Klasszikus programozás

Java nyelven

II.

Az emelt szintű

informatika érettségi

programozási feladatainak

megoldása

Eclipse (Neon, Oxygen, Photon, 2019-06)

Klement András

2016 – 2020

Utolsó tartalmi módosítás: 2020. október 31.

Tartalomjegyzék

2003 minta 1: Triatlon

Egy triatlon versenyen a versenyzőknek a verseny folyamán egymás után kell először úszniuk, kerékpározniuk majd futniuk. Az győz, aki a legrövidebb idő alatt fejezi be a versenyt.
Az egyes versenyzők adatai és időeredményei a triatlon.be fájlban találhatók.
Az első sorban 0<N<100, versenyzők száma, a következő sorokban a versenyzők adatai a következők szerint szerepelnek:

Név
Úszás idő
Kerékpár idő
Futás idő

Például:

Gipsz Jakab
1345
2312
7988

Az elért időeredményeket másodpercekben tároljuk.

1. Olvassa be a *triatlon.be* fájlból az adatokat!
2. Írja ki az összesített időeredmények alapján az első három versenyző nevét a képernyőre!
3. Írja ki a képernyőre az első helyezett nevét és azt, hogy mekkora volt az átlagsebessége (km/h-ban) az úszásban, kerékpározásban és futásban, ha a távok a következők voltak:
   Úszás: 1,5 km,
   Kerékpározás:40 km,
   Futás: 10 km!
4. Konvertálja át a versenyzők végső időeredményeit óó:pp:ss formátumra (Óra:Perc:Másodperc). Mindegyik értéket két számjeggyel jelölje!
5. A versenyzők nevét és az átkonvertált, összesített időeredményét írja ki a *triatlon.ki* fájlba!
6. A fájlban a név mellett szerepeljen az időeredmény, például: Gipsz Jakab 03:14:05!
7. Mivel a közönség kíváncsi arra is, hogy az egyes számokat (azaz az úszást, kerékpározást, futást) kik nyerték, ezért a *reszer.ki* fájlba 3 versenyző nevét és átkonvertált, időeredményét kell kiírni. Az első név az úszás győztesének a neve és az úszásban elért ideje, a második sorban a kerékpározásban győztes neve és a kerekpározásban elért időeredménye, a harmadik sorban pedig a futásban legjobb időt elért versenyző neve és időeredménye szerepeljen! Ha többen ugyanazt az eredményt érték el valamelyik versenyszámban, akkor elég az egyikük nevét kiírni.

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * Triatlon
 *
 * @author Klemand
 */
public class EmeltInfo2003mt1 {

	public static void main(String[] args) throws IOException {
		/*
		 * Kivételkezelés nem kell az emeltszintű érettségin, de ekkor a throws
		 * IOException mindig kell a fájlhasználathoz! Ügyelni kell, hogy a beolvasandó
		 * fájl a projekt főkönyvtárában legyen!
		 */

		// 1. feladat
		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása a triatlon.be fájlból \n");
		/* Mindig szövegfájlból kell beolvasni az adatokat,
		 * de a kiterjesztésük bármi lehet!
		 */


		BufferedReader beolvas = new BufferedReader(new FileReader("triatlon.be"));

		String sor = beolvas.readLine();

		/*
		 * Ha a szövegfájl UTF-8 kódolású, akkor az elején egy UTF-8 azonosító van, ami
		 * konvertálásnál hibát okoz.
		 *
		 * A probléma kezelésére két lehetőségünk van:
		 *
		 * 1) Külön olvassuk be az első sort, és ha van az elején UTF-8 azonosító,
		 * akkor elhagyjuk:
		 *
		 * if (elsoSor.charAt(0) == (char) 0xFEFF) {
		 * elsoSor = elsoSor.substring(1);
		 * }
		 *
		 * 2) Beolvasás előtt mentsük el a fájlt ANSI kódolással:
		 * szerkesztés / mentés másként / felülírás
		 *
		 *
		 * Frissítés: Ez a probléma az Eclipse 2019-06 verzióban már nincs jelen.
		 *
		 */

		int n = Integer.parseInt(sor);

		/*
		 * Az egyes versenyzők adatait külön tömbökbe olvassuk be. Az egy versenyzőhöz
		 * tartozó adatokat a tömbindex köti össze.
		 */

		String[] nev = new String[n];
		int[][] idok = new int[n][4];
		//úszás, kerékpározás, futás és összes idő
```

```
int i, j;
for (i = 1; i <= n; i++) {
        nev[i - 1] = beolvas.readLine();
        for(j = 1; j <= 3; j++) {
        idok[i - 1][j-1] = Integer.parseInt(beolvas.readLine());
        }
}
beolvas.close();

System.out.println("A 2. feladat megoldása");
int[] osszIdo = new int[n];
for (i = 1; i <= n; i++) {
        idok[i - 1][3] = 0;
        for(j=1; j<=3; j++) {
        idok[i - 1][3] += idok[i - 1][j-1];
        }
}

System.out.println("Rendezés az összidők szerint\n");
//Buborékos rendezés
String StringAsztal;
int[] intTombAsztal = new int[4];

for (i = n - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (idok[j - 1][3] > idok[j][3]) {

                        StringAsztal = nev[j - 1];
                        nev[j - 1] = nev[j];
                        nev[j] = StringAsztal;

                        intTombAsztal = idok[j - 1];
                        idok[j - 1] = idok[j];
                        idok[j] = intTombAsztal;

                }
        }
}

System.out.println("Az első három helyezett az összidők alapján: \n");
for (i = 1; i <= 3; i++) {
        System.out.println(
                        i + ". " + nev[i - 1] + " (összideje: " + idok[i - 1][3] + " s.)");
}
System.out.println("");

System.out.println("A 3. feladat megoldása");
System.out.println("Az első helyezett átlagsebessége az egyes versenyszámokban: \n");
double uszTav = 1.5; // km
double kerTav = 40; // km
double futTav = 10; // km
double uszSeb, kerSeb, futSeb;
uszSeb = 3600 * uszTav / idok[0][0];
kerSeb = 3600 * kerTav / idok[0][1];
futSeb = 3600 * futTav / idok[0][2];
System.out.println("A győztes neve: " + nev[0]);
System.out.print("Átlagsebessége úszásban: ");
System.out.printf("%.2f", uszSeb);
System.out.println(" km/h");
System.out.printf("Átlagsebessége kerékpározásban: %.2f", kerSeb);
System.out.println(" km/h");
System.out.printf("Átlagsebessége futásban: %.2f", futSeb);
System.out.println(" km/h\n");
```

```
        // 4. feladat
        System.out.println("A 4. feladat megoldása");
        System.out.println("Az összesített időeredmények átkonvertálása\n");

        String[] konvIdo = new String[n];

        for (i = 1; i <= n; i++) {
                konvIdo[i - 1] = idoKonvertalas(idok[i - 1][3]);
        }

        // 5. és 6. feladat
        System.out.println("Az 5. és 6. feladat megoldása:");
        System.out.println("Az átkonvertált időeredmények fájlba írása \n");

        PrintWriter kiir = new PrintWriter(new FileWriter("triatlon.ki"));

        for (i = 1; i <= n; i++) {
                kiir.println(nev[i - 1] + " " + konvIdo[i - 1]);
        }
        kiir.close();
        System.out.println("A fájlkiírás befejeződött. \n");

        // 7. feladat
        System.out.println("A 7. feladat megoldása:");
        System.out.println("A versenyszámok győzteseinek fájlba írása \n");

        int uszMin = 1;
        int kerMin = 1;
        int futMin = 1;
        for (i = 2; i <= n; i++) {
                if (idok[i - 1][0] < idok[uszMin -1][0]) {
                        uszMin = i;
                }
                if (idok[i - 1][1] < idok[kerMin -1][1]) {
                        kerMin = i;
                }
                if (idok[i - 1][2] < idok[futMin -1][2]) {
                        futMin = i;
                }
        }

        kiir = new PrintWriter(new FileWriter("reszer.ki"));

        kiir.println(nev[uszMin - 1] + " " + idoKonvertalas(idok[uszMin - 1][0]));
        kiir.println(nev[kerMin - 1] + " " + idoKonvertalas(idok[kerMin - 1][1]));
        kiir.println(nev[futMin - 1] + " " + idoKonvertalas(idok[futMin - 1][2]));

        kiir.close();
        System.out.println("A fájlkiírás befejeződött. \n");

}
```

```
	public static String idoKonvertalas(int ido) {
		int ora, perc, mp;
		String konv="";
		mp = ido % 60;
		perc = ido / 60;
		ora = perc / 60;
		perc = perc % 60;

		if(ora < 10) {
			konv += "0";
		}
		konv += ora + ":";

		if(perc < 10) {
			konv += "0";
		}
		konv += perc + ":";

		if(mp < 10) {
			konv += "0";
		}
		konv += mp;

		return konv; // visszaadott függvényérték
	}

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása a triatlon.be fájlból

A 2. feladat megoldása
Rendezés az összidők szerint

Az első három helyezett az összidők alapján:

1. Pál Péter (összideje: 9387 s.)
2. Kiss Pál  (összideje: 9395 s.)
3. Tóth Pál (összideje: 9460 s.)

A 3. feladat megoldása
Az első helyezett átlagsebessége az egyes versenyszámokban:

A győztes neve: Pál Péter
Átlagsebessége úszásban: 3,75 km/h
Átlagsebessége kerékpározásban: 28,82 km/h
Átlagsebessége futásban: 12,20 km/h

A 4. feladat megoldása
Az összesített időeredmények átkonvertálása

Az 5. és 6. feladat megoldása:
Az átkonvertált időeredmények fájlba írása

A fájlkiírás befejeződött.

A 7. feladat megoldása:
A versenyszámok győzteseinek fájlba írása

A fájlkiírás befejeződött.
```

A triatlon.ki szövegfájl:

```
Pál Péter 02:36:27
Kiss Pál  02:36:35
Tóth Pál 02:37:40
Tóth Ödön 02:38:34
Nagy Péter 02:38:35
Péter Pál 02:38:52
```

A reszer.ki szövegfájl:

```
Pál Péter 00:23:59
Kiss Pál  01:22:11
Tóth Ödön 00:47:12
```

2003 minta 2: Kugli

A kugli játéknál 9 bábut állítanak fel egy négyzet alakú helyre, ezeket a bábukat egy golyóval lehet ledönteni. A játékosok egymás után dobnak. A játékszabály a következő: a játékosoknak legalább annyi bábut kell ledöntenie, mint amennyit az előtte dobó játékos ledöntött. Ha kevesebbet dönt le, akkor hibapontot kap. A játékos 2 hibapont után kiesik a játékból. A játékosok száma 5, és a játék 4 kör alatt ért véget. Az első játékos az első körben nem kaphat hibapontot, de a további körökben az előző kör utolsó dobáseredményéhez viszonyítják teljesítményét.

A kugli játék körönkénti eredményeit az eredm1.txt, eredm2.txt, eredm3.txt, eredm4.txt fájlokban tároljuk. Minden fájlban a még versenyben lévő személyek nevét és a körben elért eredményét, a következőképpen:

Versenyző neve
Ledöntött bábuk száma

Ha valaki kiesett a játékból, akkor a további körök eredményeit tároló fájlban a ledöntött bábuk száma sorban értékként 10 szerepel.

Például:

Gipsz Jakab
5
Kelep Elek
10
Kiss Géza
6
Nagy Péter
4
Kovács Éva
8

1. Olvassa be az *eredm1.txt* fájlból az adatokat!
2. A szabályok alapján állapítsa meg, hogy az első körben mely játékosok kaptak hibapontot. Ezek nevét írja ki a képernyőre!
3. Olvassa be az *eredm2.txt*, *eredm3.txt* és az *eredm4.txt* fájlokból az adatokat!
4. Számítsa ki, hogy a versenyzők a játék során külön-külön mennyi bábut döntöttek le, az eredményt írja ki a képernyőre! A kiírásnál a név mellett szerepeljen az elért eredmény!
   Például:
   Gipsz Jakab 24
   Kelep Elek 12.
5. Írja ki a képernyőre a legtöbb pontot elért versenyző nevét és eredményét!
6. Állapítsa meg – a fájlokban lévő adatok alapján – a kiesett versenyzők nevét és azt, hogy melyik körben estek ki! A neveket és a kör számát írja ki a képernyőre!
7. Írja ki a képernyőre azoknak a versenyzőknek a nevét, akiknek sikerült 9 bábut eldönteniük a dobásukkal a játék során! A nevük mellett szerepeljen, hogy mely körökben érték el ezt az eredményt! (A név legfeljebb egyszer szerepeljen!)

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

/**
 * Kugli
 *
 * @author Klemand
 */
public class EmeltInfo2003mt2 {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása az eredm1.txt fájlból \n");
		int n = 5; // A feladatban megadott konstans
		String[] eredm = new String[4];
		eredm[0] = "eredm1.txt";
		eredm[1] = "eredm2.txt";
		eredm[2] = "eredm3.txt";
		eredm[3] = "eredm4.txt";

		BufferedReader beolvas = new BufferedReader(new FileReader(eredm[0]));

		String[] nev = new String[n];// játékos
		int[][] pont = new int[4][n];// kör, játékos
		int i;
		for (i = 1; i <= n; i++) {
			nev[i - 1] = beolvas.readLine();
			pont[0][i - 1] = Integer.parseInt(beolvas.readLine());
			// a ledöntött bábuk száma az 1. körben
		}

		beolvas.close();

		System.out.println("A 2. feladat megoldása");
		System.out.println("Hibapontok az első körben:");
		int[] hibaPont = new int[n];

		for (i = 2; i <= n; i++) {
			if (pont[0][i - 1] < pont[0][i - 2]) {
				hibaPont[i - 1]++;
			}
		}
		for (i = 2; i <= n; i++) {
			if (hibaPont[i - 1] > 0) {
				System.out.println(nev[i - 1]);
			}
		}
		System.out.println("");

		System.out.println("A 3. feladat megoldása");
		System.out.println("Az adatok beolvasása a többi txt fájlból \n");

		int k;
		for (k = 2; k <= 4; k++) {

			beolvas = new BufferedReader(new FileReader(eredm[k - 1]));

			for (i = 1; i <= n; i++) {
				beolvas.readLine(); // A nevet most már csak átugorjuk
				pont[k - 1][i - 1] = Integer.parseInt(beolvas.readLine());
			}

			beolvas.close();
		}
```

```
System.out.println("A 4. feladat megoldása");
System.out.println("Ledöntött bábuk száma:");

int[] osszPont = new int[n];

for (i = 1; i <= n; i++) {
        for (k = 1; k <= 4; k++) {
                if (pont[k - 1][i - 1] < 10) {
                        osszPont[i - 1] += pont[k - 1][i - 1];
                }
        }
}

for (i = 1; i <= n; i++) {
        System.out.println(nev[i - 1] + " " + osszPont[i - 1]);
}
System.out.println("");

System.out.println("Az 5. feladat megoldása");
System.out.println("A legtöbb pontot elért versenyző:");

int max = 1;
for (i = 2; i <= n; i++) {
        if (osszPont[i - 1] > osszPont[max - 1]) {
                max = i;
        }
}
System.out.println(nev[max - 1] + " " + osszPont[max - 1] + "\n");

System.out.println("A 6. feladat megoldása");
System.out.println("A kiesett versenyzők és a kiesés köre:");

int[] kieses = new int[n];
for (i = 1; i <= n; i++) {
        kieses[i - 1] = -1; // irreális kezdőérték
}

for (i = 1; i <= n; i++) {
        k = 2;
        while (k <= 4 && pont[k - 1][i - 1] < 10) {
                k++;
        }
        if (k <= 4) {
                kieses[i - 1] = k - 1;
        }
}

for (i = 1; i <= n; i++) {
        if (kieses[i - 1] > -1) {
                System.out.println(nev[i - 1] + " " + kieses[i - 1]);
        }
}
System.out.println("");
```

```
                System.out.println("A 7. feladat megoldása");
                System.out.println("A 9 pontos versenyzők és a 9 pontos körök:");

                boolean[] kilences = new boolean[n];
                for (i = 1; i <= n; i++) {
                        kilences[i - 1] = false; // Ha lesz kilencese, igazra állítjuk.
                }

                for (i = 1; i <= n; i++) {
                        for (k = 1; k <= 4; k++) {
                                if (pont[k - 1][i - 1] == 9) {
                                        kilences[i - 1] = true;
                                }
                        }
                }

                for (i = 1; i <= n; i++) {
                        if (kilences[i - 1]) {
                                System.out.print(nev[i - 1]);
                                for (k = 1; k <= 4; k++) {
                                        if (pont[k - 1][i - 1] == 9) {
                                                System.out.print(" " + k);
                                        }
                                }
                                System.out.println();
                        }
                }
                System.out.println("");
        }

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása az eredm1.txt fájlból

A 2. feladat megoldása
Hibapontok az első körben:
Kiss Géza
Kovács Éva

A 3. feladat megoldása
Az adatok beolvasása a többi txt fájlból

A 4. feladat megoldása
Ledöntött bábuk száma:
Gipsz Jakab 20
Kelep Elek 32
Kiss Géza 12
Nagy Péter 33
Kovács Éva 14

Az 5. feladat megoldása
A legtöbb pontot elért versenyző:
Nagy Péter 33

A 6. feladat megoldása
A kiesett versenyzők és a kiesés köre:
Gipsz Jakab 3
Kiss Géza 2
Kovács Éva 2

A 7. feladat megoldása
A 9 pontos versenyzők és a 9 pontos körök:
Kelep Elek 1 4
Nagy Péter 2 4
```

2004 május: Személyazonosító jel

Az ország állampolgárainak van egyedi azonosítójuk. Ez a személyazonosító jel.
Az 1997. január 1-je után születetteknél ez a következőképpen néz ki.

A személyazonosító jel 11 jegyű.
Az első jegy a személy nemét jelöli, az alábbi táblázat alapján.

1997. január 1. és 1999. december 31. között született		1999. december 31. után született	
férfi	nő	férfi	nő
1	2	3	4

A 2–7 számjegyek a születési év utolsó két jegyét, a születési hónapot és napot tartalmazza.
A 8–10. számjegyek az azonos napon születettek születési sorszáma. A 11. jegy az első tíz jegyből képzett ellenőrző szám.

Írjon olyan programot, amely végrehajtja az alábbi utasításokat!

a) Kérje be egy személyazonosító jel első 10 jegyét!

b) Írassa ki a képernyőre, a személyazonosító jel alapján, hogy az adott személy férfi vagy nő!

c) Írassa ki a képernyőre, az adott személy születési sorszámát!

d) Írassa ki a képernyőre, hogy hányadik születésnapja van ebben az évben a személynek!

e) Kérjen be egy másik személyazonosító jelet is! (Szintén csak az első 10 jegyét!)

f) Határozza meg, a két beadott személyazonosító jel alapján, hogy melyik személy idősebb! (Ha két ember ugyanakkor született, akkor a 8–10. jegy alapján döntse el, melyik az idősebb!) Az eredményt a képernyőn jelenítse meg!

g) Mennyi a különbség a születési éveik között? Figyeljen a 1999. dec. 31. után születettekre is! Az eredményt írassa ki a képernyőre!

h) A másodikként beadott személyazonosító jelből, számítsa ki a 11. jegyet és írassa ki a képernyőre a teljes személyazonosító jelet. A számítás a következő szabály alapján működik.
A első tíz számjegy mindegyikét szorozzuk meg egy számmal. Mégpedig a 10. helyen állót eggyel, a 9. helyen állót kettővel és így tovább. Az így kapott szorzatokat adjuk össze. A kapott összeget osszuk el tizeneggyel. Az osztás maradéka lesz a 11. jegy.
Kivéve, ha a maradék 10. Mert ekkor azt a születési sorszámot nem adják ki. Ebben az esetben írja ki, hogy hibás a születési sorszám!

i) Mindkét korábban beadott személyazonosító jel első 10 jegyét írja a `szemszam.txt` fájlba!

```java
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Calendar;
import java.util.Scanner;

/**
 * Személyazonosító jel
 *
 * @author Klemand
 */
public class EmeltInfo2004maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az a) feladat megoldása:");
		System.out.print("Kérem az első személyi szám első tíz jegyét! ");

		Scanner sc = new Scanner(System.in);

		String kod1 = sc.nextLine();
		System.out.println();

		System.out.println("A b) feladat megoldása:");
		if (kod1.substring(0, 1).equals("1") || kod1.substring(0, 1).equals("3")) {
			System.out.println("Az adott személy férfi.");
		} else {
			System.out.println("Az adott személy nő.");
		}
		System.out.println("");

		System.out.println("A c) feladat megoldása:");
		System.out.println("Az adott személy születési sorszáma: " + kod1.substring(7, 10));
		System.out.println("");

		int szulEv1, aktEv;
		System.out.println("A d) feladat megoldása:");
		szulEv1 = szulEv(kod1);

		Calendar cal = Calendar.getInstance();
		aktEv = cal.get(Calendar.YEAR);
		System.out.println("Az adott személy születési éve: " + szulEv1 + ", az idei év: " + aktEv);
		System.out.println("Tehát idén " + (aktEv - szulEv1) + ". születésnapja van.");
		System.out.println("");

		System.out.println("Az e) feladat megoldása:");
		System.out.print("Kérem a második személyi szám első tíz jegyét! ");

		String kod2 = sc.nextLine();

		sc.close();
		System.out.println();

		System.out.println("Az f) feladat megoldása:");
		int szulEv2;
		szulEv2 = szulEv(kod2);
		String szul1 = "" + szulEv1 + kod1.substring(3, 10);
		String szul2 = "" + szulEv2 + kod2.substring(3, 10);
		/*
		 * Ha számmal kezdődik az összegzés, de Stringet szeretnénk eredményül, üres
		 * String-gel kezdjük az összeadást.
		 */
```

```
                int i = 1;
                while (i <= 10 && szul1.substring(i - 1, i).equals(szul2.substring(i - 1, i))) {
                        i++;
                }
                if (i <= 10) {
                        if (szul1.substring(i - 1, i).compareTo(szul2.substring(i - 1, i)) < 0) {
                                System.out.println("Az első személy az idősebb");
                        } else {
                                System.out.println("A második személy az idősebb");
                        }
                } else {
                        System.out.println("A két személyi szám azonos!");
                }

                System.out.println("");

                System.out.println("A g) feladat megoldása:");

                System.out.println("A születési éveik közti különbség: " + (Math.abs(szulEv2 - szulEv1)));
                System.out.println("");

                System.out.println("A h) feladat megoldása:");
                int osszeg = 0;
                for (i = 10; i >= 1; i--) {
                        osszeg += Integer.parseInt(kod2.substring(i - 1, i)) * (11 - i);
                }

                int jegy = osszeg % 11;
                if (jegy != 10) {
                        System.out.print("A teljes második személyi szám: ");

                        System.out.println(kod2 + jegy);
                        // String + int eredménye összefűzés

                } else {
                        System.out.println("Hibás a születési sorszám!");
                }
                System.out.println("");

                System.out.println("Az i) feladat megoldása:");

                System.out.println("A személyi számok első 10 jegyének kiíratása a szemszam.txt fájlba.");

                PrintWriter kiir = new PrintWriter(new FileWriter("szemszam.txt"));

                kiir.println(kod1);
                kiir.println(kod2);

                kiir.close();

                System.out.println("A fájlkiírás befejeződött.");
                System.out.println("");
        }

        public static int szulEv(String kod) {
                if (kod.substring(0, 1).equals("1") || kod.substring(0, 1).equals("2")) {
                        return 1900 + Integer.parseInt(kod.substring(1, 3));
                } else {
                        return 2000 + Integer.parseInt(kod.substring(1, 3));
                }
        }

}
```

```
Az a) feladat megoldása:
Kérem az első személyi szám első tíz jegyét! 1111111111

A b) feladat megoldása:
Az adott személy férfi.

A c) feladat megoldása:
Az adott személy születési sorszáma: 111

A d) feladat megoldása:
Az adott személy születési éve: 1911, az idei év: 2018
Tehát idén 107. születésnapja van.

Az e) feladat megoldása:
Kérem a második személyi szám első tíz jegyét! 2121212121

Az f) feladat megoldása:
Az első személy az idősebb

A g) feladat megoldása:
A születési éveik közti különbség: 1

A h) feladat megoldása:
A teljes második személyi szám: 21212121218

Az i) feladat megoldása:
A személyi számok első 10 jegyének kiíratása a szemszam.txt fájlba.
A fájlkiírás befejeződött.
```

A szemszam.txt szövegfájl:

```
1111111111
2121212121
```

2005 május: Lottó

Magyarországon 1957 óta lehet ötös lottót játszani. A játék lényege a következő: a lottószelvényeken 90 szám közül 5 számot kell a fogadónak megjelölnie. Ha ezek közül 2 vagy annál több megegyezik a kisorsolt számokkal, akkor nyer. Az évek során egyre többen hódoltak ennek a szerencsejátéknak és a nyeremények is egyre nőttek.

Adottak a *lottosz.dat* szöveges állományban a 2003. év 51 hetének ötös lottó számai. Az első sorában az első héten húzott számok vannak, szóközzel elválasztva, a második sorban a második hét lottószámai vannak stb.

Például:
```
37 42 44 61 62
18 42 54 83 89
...
9 20 21 59 68
```

A lottószámok minden sorban emelkedő számsorrendben szerepelnek.

Az állományból kimaradtak az 52. hét lottószámai. Ezek a következők voltak: 89 24 34 11 64.

Készítsen programot a következő feladatok megoldására!

1. Kérje be a felhasználótól az 52. hét megadott lottószámait!
2. A program rendezze a bekért lottószámokat emelkedő sorrendbe!
   A rendezett számokat írja ki a képernyőre!
3. Kérjen be a felhasználótól egy egész számot 1-51 között! A bekért adatot nem kell ellenőrizni!
4. Írja ki a képernyőre a bekért számnak megfelelő sorszámú hét lottószámait, a *lottosz.dat* állományban lévő adatok alapján!
5. A *lottosz.dat* állományból beolvasott adatok alapján döntse el, hogy volt-e olyan szám, amit egyszer sem húztak ki az 51 hét alatt! A döntés eredményét (Van/Nincs) írja ki a képernyőre!
6. A *lottosz.dat* állományban lévő adatok alapján állapítsa meg, hogy hányszor volt páratlan szám a kihúzott lottószámok között! Az eredményt a képernyőre írja ki!
7. Fűzze hozzá a *lottosz.dat* állományból beolvasott lottószámok után a felhasználótól bekért, és rendezett 52. hét lottószámait, majd írja ki az összes lottószámot a *lotto52.ki* szöveges fájlba! A fájlban egy sorba egy hét lottószámai kerüljenek, szóközzel elválasztva egymástól!
8. Határozza meg a *lotto52.ki* állomány adatai alapján, hogy az egyes számokat hányszor húzták ki 2003-ban. Az eredményt írja ki a képernyőre a következő formában: az első sor első eleme az a szám legyen ahányszor az egyest kihúzták! Az első sor második eleme az az érték legyen, ahányszor a kettes számot kihúzták stb.! (Annyit biztosan tudunk az értékekről, hogy mindegyikük egyjegyű.)

Példa egy lehetséges eredmény elrendezésére (6 sorban, soronként 15 érték).

4 2 2 4 2 2 6 1 1 2 1 5 2 1 1
1 3 5 0 5 5 2 6 6 5 1 0 6 4 3
3 3 5 4 3 1 4 2 2 4 2 4 1 2 3
4 2 1 2 3 2 2 2 4 4 5 1 3 5 5
5 2 0 2 2 4 4 3 1 3 6 1 5 6 2
4 3 2 2 3 1 1 4 1 3 3 2 1 5 3

9. Adja meg, hogy az 1-90 közötti prímszámokból melyiket nem húzták ki egyszer sem az elmúlt évben. A feladat megoldása során az itt megadott prímszámokat felhasználhatja vagy előállíthatja! (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89.)

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Lottó
 *
 * @author Klemand
 */
public class EmeltInfo2005maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása:");

		System.out.println("Kérem az 52. hét megadott lottószámait (89 24 34 11 64) szóközökkel elválasztva!");
		// Feltételezhetjük, hogy valóban ezt az öt számot írjuk be.

		Scanner sc = new Scanner(System.in);

		String sor;
		String[] daraboltSor;

		sor = sc.nextLine().trim();
		//Az esetleges szóközöket le kell vágni a sor széleiről!

		daraboltSor = sor.split(" ");
		int[] het52 = new int[5];

		int i;

		for (i = 1; i <= 5; i++) {
			het52[i - 1] = Integer.parseInt(daraboltSor[i - 1]);
		}

		System.out.println("");

		System.out.println("A 2. feladat megoldása:");
		// Buborékos rendezés
		int j, asztal;

		for (i = 4; i >= 1; i--) {
			for (j = 1; j <= i; j++) {
				if (het52[j - 1] > het52[j]) {
					asztal = het52[j - 1];
					het52[j - 1] = het52[j];
					het52[j] = asztal;
				}
			}
		}

		System.out.print("Az 52. hét lottószámai nagyság szerint rendezve: ");
		for (i = 1; i <= 5; i++) {
			System.out.print(het52[i - 1] + " ");
		}
		System.out.println("\n");

		System.out.println("A 3. feladat megoldása:");
		System.out.print("Kérem a vizsgált hét sorszámát 1 és 51 között! ");

		int het = sc.nextInt();
		sc.close();
		System.out.println();
```

```
System.out.println("A 4. feladat megoldása:");

BufferedReader beolvas = new BufferedReader(new FileReader("lottosz.dat"));

int[][] lottoSzamok = new int[52][5];

for (i = 1; i <= 51; i++) {
        sor = beolvas.readLine();
        daraboltSor = sor.split(" ");
        for (j = 1; j <= 5; j++) {
                lottoSzamok[i - 1][j - 1] = Integer.parseInt(daraboltSor[j - 1]);
        }
}

beolvas.close();

System.out.print("A(z) " + het + ". hét nyerőszámai: ");

for (j = 1; j <= 5; j++) {
        System.out.print(lottoSzamok[het - 1][j - 1] + " ");
}
System.out.println("\n");

System.out.println("Az 5. feladat megoldása:");

int[] lotto = new int[90];
// A lotto tömbben tároljuk, hogy melyik számot hányszor húzták ki.

for (i = 1; i <= 90; i++) {
        lotto[i - 1] = 0;
}

int kihuzott;

for (i = 1; i <= 51; i++) {
        for (j = 1; j <= 5; j++) {
                kihuzott = lottoSzamok[i - 1][j - 1];
                lotto[kihuzott - 1]++;
        }
}

i = 1;
while ((i <= 90) && (lotto[i - 1] > 0)) {
        i++;
}
if (i <= 90) {
        System.out.println("Volt olyan szám, amit egyszer sem húztak ki, pl.: " + i + "\n");
} else {
        System.out.println("Nem volt olyan szám, amit egyszer sem húztak ki.\n");
}

System.out.println("Az 6. feladat megoldása:");

// Hivatalos megoldás: Hány páratlan szám volt összesen?
int paratlanDb = 0;
for (i = 1; i <= 51; i++) {
        for (j = 1; j <= 5; j++) {
                if (lottoSzamok[i - 1][j - 1] % 2 == 1) {
                        paratlanDb++;
                }
        }
}
System.out.println("Összesen " + paratlanDb + " esetben volt a nyertes szám páratlan.");
System.out.println("");
```

```
/*
 * Alternatív értelmezés: Hány héten szerepelt páratlan szám a kihúzottak
 * között?
 */

int paratlanHet = 0;
for (i = 1; i <= 51; i++) {
        j = 1;
        while ((j <= 5) && (lottoSzamok[i - 1][j - 1] % 2 == 0)) {
                j++;
        }
        if (j <= 5) {
                paratlanHet++;
        }
}
System.out.println(paratlanHet + " héten volt a nyertes számok között páratlan.");
System.out.println("");

System.out.println("A 7. feladat megoldása:");

for (j = 1; j <= 5; j++) {
        lottoSzamok[51][j - 1] = het52[j - 1];
}

PrintWriter kiir = new PrintWriter(new FileWriter("lotto52.ki"));

for (i = 1; i <= 52; i++) {
        for (j = 1; j <= 5; j++) {
                kiir.print(lottoSzamok[i - 1][j - 1] + " ");
        }
        kiir.println();
}

kiir.close();
System.out.println("A fájlkiírás befejeződött.");
System.out.println("");

System.out.println("A 8. feladat megoldása: ");

/*
 * 51 hétre már megvan a lotto tömbben, csak ki kell egészíteni az 52. héttel
 */

for (j = 1; j <= 5; j++) {
        kihuzott = lottoSzamok[51][j - 1];
        lotto[kihuzott - 1]++;
}

for (i = 1; i <= 90; i++) {
        System.out.print(lotto[i - 1] + " ");
        if (i % 15 == 0) {
                System.out.println("");
        }
}
System.out.println("");
```

```
                // 9. feladat
                // Egyszer sem kihúzott prímek
                System.out.println("A 9. feladat megoldása:");
                System.out.print("Egyszer sem kihúzott prímek: ");

                int[] primek = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
                                 79, 83, 89 };
                int db = primek.length;
                int prim;
                for (i = 1; i <= db; i++) {
                        prim = primek[i - 1];
                        if (lotto[prim - 1] == 0) {
                                System.out.print(prim + " ");
                        }
                }
                System.out.println("\n");
        }

}
```

```
Az 1. feladat megoldása:
Kérem az 52. hét megadott lottószámait (89 24 34 11 64) szóközökkel elválasztva!
 89 24 34 11 64

A 2. feladat megoldása:
Az 52. hét lottószámai nagyság szerint rendezve: 11 24 34 64 89

A 3. feladat megoldása:
Kérem a vizsgált hét sorszámát 1 és 51 között! 7

A 4. feladat megoldása:
A(z) 7. hét nyerőszámai: 21 29 37 48 68

Az 5. feladat megoldása:
Volt olyan szám, amit egyszer sem húztak ki, pl.: 11

Az 6. feladat megoldása:
Összesen 126 esetben volt a nyertes szám páratlan.

49 héten volt a nyertes számok között páratlan.

A 7. feladat megoldása:
A fájlkiírás befejeződött.

A 8. feladat megoldása:
4 2 2 4 2 2 6 1 1 2 1 5 2 1 1
1 3 5 0 5 5 2 6 6 5 1 0 6 4 3
3 3 5 4 3 1 4 2 2 4 2 4 1 2 3
4 2 1 2 3 2 2 2 4 4 5 1 3 5 5
5 2 0 2 2 4 4 3 1 3 6 1 5 6 2
4 3 2 2 3 1 1 4 1 3 3 2 1 5 3

A 9. feladat megoldása:
Egyszer sem kihúzott prímek: 19
```

A lotto.ki szöveges fájl:

```
37 42 44 61 62
18 42 54 83 89
5 12 31 53 60
1 28 47 56 70
54 56 57 59 71
7 21 33 39 86
21 29 37 48 68
10 21 29 40 87
13 33 73 77 78
2 23 65 71 84
3 21 28 30 33
23 31 42 73 85
4 23 42 61 64
17 60 66 71 85
12 60 66 67 72
46 50 58 62 76
20 32 43 65 73
55 56 58 61 71
18 38 41 67 89
32 41 59 66 79
25 35 37 74 86
1 45 60 61 82
7 20 35 58 83
7 37 40 46 51
2 6 47 74 80
1 5 22 44 88
23 33 34 71 89
4 56 74 77 89
17 18 51 52 75
7 29 30 77 80
17 18 28 35 90
6 24 25 53 79
7 12 18 38 90
25 28 45 55 74
10 29 60 74 86
7 24 25 50 76
20 40 52 54 90
16 30 81 83 87
20 22 23 50 67
59 68 75 80 85
32 45 55 70 78
13 40 55 56 76
3 14 24 73 83
23 25 28 66 76
24 33 34 39 54
12 28 34 61 70
1 4 8 69 74
4 15 46 49 59
24 31 67 71 73
12 26 36 46 49
9 20 21 59 68
11 24 34 64 89
```

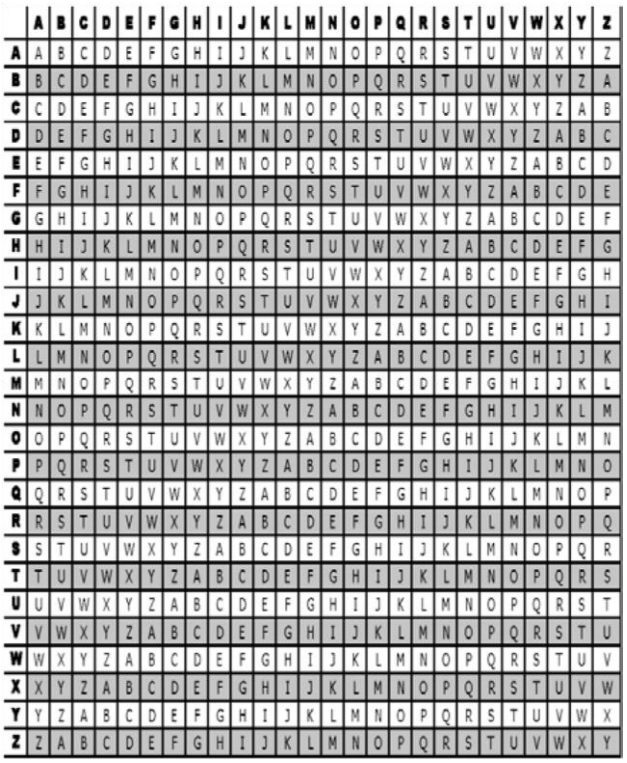
2005 október: Vigenère tábla

Már a XVI. században komoly titkosítási módszereket találtak ki az üzenetek elrejtésére. A század egyik legjobb kriptográfusának Blaise de Vigenère-nek a módszerét olvashatja a következőkben.

A kódoláshoz egy táblázatot és egy ún. kulcsszót használt. A táblázatot a jobb oldali ábra tartalmazza.

A tábla adatait a `vtabla.dat` fájlban találja a következő formában.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA
CDEFGHIJKLMNOPQRSTUVWXYZAB
DEFGHIJKLMNOPQRSTUVWXYZABC
EFGHIJKLMNOPQRSTUVWXYZABCD
FGHIJKLMNOPQRSTUVWXYZABCDE
```



Készítsen programot `kodol` néven a következő feladatok végrehajtására!

1. Kérjen be a felhasználótól egy maximum 255 karakternyi, nem üres szöveget!
   A továbbiakban ez a nyílt szöveg.
2. Alakítsa át a nyílt szöveget, hogy a későbbi kódolás feltételeinek megfeleljen!

A kódolás feltételei:

- A magyar ékezetes karakterek helyett ékezetmenteseket kell használni.
  (Például á helyett a; ő helyett o stb.)
- A nyílt szövegben az átalakítás után csak az angol ábécé betűi szerepelhetnek.
- A nyílt szöveg az átalakítás után legyen csupa nagybetűs.

3. Írja ki a képernyőre az átalakított nyílt szöveget!
4. Kérjen be a felhasználótól egy maximum 5 karakteres, nem üres kulcsszót!
   A kulcsszó a kódolás feltételeinek megfelelő legyen! (Sem átalakítás, sem ellenőrzés nem kell!)
   Alakítsa át a kulcsszót csupa nagybetűssé!
5. A kódolás első lépéseként fűzze össze a kulcsszót egymás után annyiszor, hogy az így kapott karaktersorozat (továbbiakban kulcsszöveg) hossza legyen egyenlő a kódolandó szöveg hosszával! Írja ki a képernyőre az így kapott kulcsszöveget!
6. A kódolás második lépéseként a következőket hajtsa végre! Vegye az átalakított nyílt szöveg első karakterét, és keresse meg a `vtabla.dat` fájlból beolvasott táblázat első oszlopában! Ezután vegye a kulcsszöveg első karakterét, és keresse meg a táblázat első sorában! Az így kiválasztott sor és oszlop metszéspontjában lévő karakter lesz a kódolt szöveg első karaktere. Ezt ismételje a kódolandó szöveg többi karakterével is!
7. Írja ki a képernyőre és a `kodolt.dat` fájlba a kapott kódolt szöveget!

Példa:

Nyílt szöveg: `Ez a próba szöveg, amit kódolunk!`

Szöveg átalakítása: `EZAPROBASZOVEGAMITKODOLUNK`

Kulcsszó: `auto`

Kulcsszó nagybetűssé alakítása: `AUTO`

Nyílt szöveg és kulcsszöveg együtt:

E	Z	A	P	R	O	B	A	S	Z	O	V	E	G	A	M	I	T	K	O	D	O	L	U	N	K
A	U	T	O	A	U	T	O	A	U	T	O	A	U	T	O	A	U	T	O	A	U	T	O	A	U

Kódolt szöveg:

E	T	T	D	R	I	U	O	S	T	H	J	E	A	T	A	I	N	D	C	D	I	E	I	N	E

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Vignère tábla
 *
 * @author user
 */

public class EmeltInfo2005okt {

        public static void main(String[] args) throws IOException {

                System.out.println("Az 1. feladat megoldása");
                System.out.println("Ékezetes nyílt szöveg beolvasása a billentyűzetről \n");
                System.out.println("Kérek egy max. 255 karakterből álló ékezetes szöveget!");
                Scanner sc = new Scanner(System.in);

                String nyiSz = sc.nextLine();// nyílt szöveg
                System.out.println();

                System.out.println("A 2. feladat megoldása");
                System.out.println("A nyílt szöveg átalakítása \n");
                nyiSz = nyiSz.toUpperCase();
                String abc = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
                String aSz = "";// átalakított szöveg
                String betu;

                int i;

                for (i = 1; i <= nyiSz.length(); i++) {
                        betu = nyiSz.substring(i - 1, i);
                        if (abc.contains(betu)) {
                                aSz += betu;
                        }
                        if (betu.equals("Á")) {
                                aSz += "A";
                        }
                        if (betu.equals("É")) {
                                aSz += "E";
                        }
                        if (betu.equals("Í")) {
                                aSz += "I";
                        }
                        if (betu.equals("Ó") || betu.equals("Ö") || betu.equals("Ő")) {
                                aSz += "O";
                        }
                        if (betu.equals("Ú") || betu.equals("Ü") || betu.equals("Ű")) {
                                aSz += "U";
                        }
                }

                System.out.println("A 3. feladat megoldása");
                System.out.println("Az átalakított nyílt szöveg kiírása \n");
                System.out.println("Az átalakított szöveg: \n" + aSz);

                System.out.println("\nA 4. feladat megoldása");
                System.out.println("A kulcsszó beolvasása a billentyűzetről \n");
                System.out.println("Kérek egy max. 5 karakterből álló ékezetmentes szót!");
                String kulcs = sc.nextLine();
                sc.close();
```

```
kulcs = kulcs.toUpperCase();
System.out.println("A nagybetűssé alakított kulcsszó: \n" + kulcs);

System.out.println("\nAz 5. feladat megoldása");
System.out.println("A kulcsszó-füzér megalkotása");

int db = aSz.length() / kulcs.length() + 1;
String kulcsFuzer = "";

for (i = 1; i <= db; i++) {
        kulcsFuzer += kulcs;
}
kulcsFuzer = kulcsFuzer.substring(0, aSz.length());

System.out.println("A kulcsszó-füzér: \n" + kulcsFuzer);

System.out.println("\nA 6. feladat megoldása");
System.out.println("Fájlbeolvasás és a kódolás végrehajtása \n");

String[][] vTabla = new String[26][26];
// Tudjuk, hogy a Vignère tábla konstans 26x26 méretű

BufferedReader beolvas = new BufferedReader(new FileReader("Vtabla.dat"));

int j;

String fSor;
for (i = 1; i <= 26; i++) {
        fSor = beolvas.readLine();
        for (j = 1; j <= 26; j++) {
                vTabla[i - 1][j - 1] = fSor.substring(j - 1, j);
        }
}

beolvas.close();

String koSz;// kódolt szöveg
koSz = "";

int n;
for (n = 1; n <= aSz.length(); n++) {
        // A tábla sorának meghatározása
        i = abc.indexOf(aSz.substring(n - 1, n)) + 1;

        // A tábla oszlopának meghatározása
        j = abc.indexOf(kulcsFuzer.substring(n - 1, n)) + 1;

        koSz += vTabla[i - 1][j - 1];
}

System.out.println("A 7. feladat megoldása");
System.out.println("A kódolt szöveg kiírása a képernyőre és szövegfájlba:");
System.out.println(koSz);

PrintWriter kiir = new PrintWriter(new FileWriter("kodolt.dat"));
kiir.println(koSz);
kiir.close();
System.out.println("A fájlkiírás befejeződött. \n");
```

```
                System.out.println("Ellenőrzés: visszafejtés");// Nem része a feladatnak!
                System.out.println("A  nyílt szöveg visszafejtése a kódolt szövegből:");
                String viSz;
                viSz = "";

                for (n = 1; n <= aSz.length(); n++) {
                        // A tábla oszlopát a kulcsfüzér aktuális betűje határozza meg
                        j = abc.indexOf(kulcsFuzer.substring(n - 1, n)) + 1;
                        /*
                         * Ebből az oszlopból kell kiválasztanunk a titkos szöveg aktuális betűjét,
                         * megkeresni, hogy melyik sorban van, annak az első eleme kell.
                         */
                        i = 1;
                        while (!(koSz.substring(n - 1, n).equals(vTabla[i - 1][j - 1]))) {
                                i++;
                        }
                        viSz += vTabla[i - 1][0];
                }

                System.out.println(viSz);
        }

}
```

```
Az 1. feladat megoldása
Ékezetes nyílt szöveg beolvasása a billentyűzetről

Kérek egy max. 255 karakterből álló ékezetes szöveget!
A hűsítő tó fölé ezüst holdsugár kúszik.

A 2. feladat megoldása
A nyílt szöveg átalakítása

A 3. feladat megoldása
Az átalakított nyílt szöveg kiírása

Az átalakított szöveg:
AHUSITOTOFOLEEZUSTHOLDSUGARKUSZIK

A 4. feladat megoldása
A kulcsszó beolvasása a billentyűzetről

Kérek egy max. 5 karakterből álló ékezetmentes szót!
hold
A nagybetűssé alakított kulcsszó:
HOLD

Az 5. feladat megoldása
A kulcsszó-füzér megalkotása
A kulcsszó-füzér:
HOLDHOLDHOLDHOLDHOLDHOLDHOLDHOLDH

A 6. feladat megoldása
Fájlbeolvasás és a kódolás végrehajtása

A 7. feladat megoldása
A kódolt szöveg kiírása a képernyőre és szövegfájlba:
HVFVPHZWVTZOLSKXZHSRSRDXNOCNBGKLR
A fájlkiírás befejeződött.

Ellenőrzés: visszafejtés
A  nyílt szöveg visszafejtése a kódolt szövegből:
AHUSITOTOFOLEEZUSTHOLDSUGARKUSZIK


A kodolt.dat szövegfájl

HVFVPHZWVTZOLSKXZHSRSRDXNOCNBGKLR
```

2006 február: Telefonszámla

Egy új szolgáltatás keretében ki lehet kérni a napi telefonbeszélgetéseink listáját. A listát egy fájlban küldik meg, amelyben a következő adatok szerepelnek: hívás kezdete, hívás vége, hívott telefonszám. A hívás kezdete és vége óra, perc, másodperc formában szerepel.

Például:

6 15 0 6 19 0
395682211

9 58 15 10 3 53
114571155

Óra	perc	mperc	Óra	perc	mperc
Telefonszám					
Óra	perc	mperc	Óra	perc	mperc
Telefonszám					

A hívások listája időben rendezett módon tartalmazza az adatokat, és szigorúan csak egy napi adatot, azaz nincsenek olyan beszélgetések, amelyeket előző nap kezdtek vagy a következő napon fejeztek be. Továbbá az elmúlt időszak statisztikái alapján tudjuk, hogy a napi hívások száma nem haladja meg a kétszázat.

A telefonálás díjait a következő táblázat foglalja össze.

Hívásirány	Csúcsidőben $7^{\underline{00}}$ - $18^{\underline{00}}$ (Ft/perc)	Csúcsidőn kívül $0^{\underline{00}}$ – $7^{\underline{00}}$ és $18^{\underline{00}}$ – $24^{\underline{00}}$ (Ft/perc)
Vezetékes	30	15
Mobil társaság	69,175	46,675

További fontos információk:

- A csúcsidő reggel 7:00:00-kor, a csúcsidőn kívüli időszak pedig 18:00:00-kor kezdődik.
  A díjazás számításakor az számít, hogy mikor kezdte az illető a beszélgetést.
  (Például: ha 17:55-kor kezdett egy beszélgetést, de azt 18:10-kor fejezte be, akkor is csúcsidőbeli díjakkal kell számlázni.)
- Minden megkezdett perc egy egész percnek számít.
- Minden telefonszám elején egy kétjegyű körzetszám, illetve mobil hívószám található. A mobil hívószámok: 39, 41, 71 kezdődnek, minden egyéb szám vezetékes hívószámnak felel meg.

A következő feladatokat oldja meg egy program segítségével! A programot mentse `szamla` néven!

1. Kérjen be a felhasználótól egy telefonszámot! Állapítsa meg a program segítségével, hogy a telefonszám mobil-e vagy sem! A megállapítást írja ki a képernyőre!
2. Kérjen be továbbá egy hívás kezdeti és hívás vége időpontot óra perc másodperc formában! A két időpont alapján határozza meg, hogy a számlázás szempontjából hány perces a beszélgetés! A kiszámított időtartamot írja ki a képernyőre!

3. Állapítsa meg a *hivasok.txt* fájlban lévő hívások időpontja alapján, hogy hány számlázott percet telefonált a felhasználó hívásonként! A kiszámított számlázott perceket írja ki a *percek.txt* fájlba a következő formában!

   ```
   perc telefonszám
   ```

4. Állapítsa meg a *hivasok.txt* fájl adatai alapján, hogy hány hívás volt csúcsidőben és csúcsidőn kívül! Az eredményt jelenítse meg a képernyőn!

5. A *hivasok.txt* fájlban lévő időpontok alapján határozza meg, hogy hány percet beszélt a felhasználó mobil számmal és hány percet vezetékessel! Az eredményt jelenítse meg a képernyőn!

6. Összesítse a *hivasok.txt* fájl adatai alapján, mennyit kell fizetnie a felhasználónak a csúcsdíjas hívásokért! Az eredményt a képernyőn jelenítse meg!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Telefonszámla
 *
 * @author Klemand
 */

public class EmeltInfo2006feb {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Egy billentyűzetről bekért telefonszám jellegének meghatározása");

		Scanner sc = new Scanner(System.in);

		System.out.print("Kérem a telefonszámot: ");
		String telefonSzam = sc.nextLine();

		String jelleg = jellegMeghat(telefonSzam);

		System.out.println("A megadott telefonszám: " + jelleg + "\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("A hívásidő kiszámítása a bekért időpontok alapján");

		System.out.println("Kérem a hívás kezdetének időpontját óra perc másodperc formátumban");
		System.out.print("szóközökkel elválasztva: ");
		String ido = sc.nextLine();
		String[] daraboltIdo = ido.split(" ");
		int ora = Integer.parseInt(daraboltIdo[0]);
		int perc = Integer.parseInt(daraboltIdo[1]);
		int mp = Integer.parseInt(daraboltIdo[2]);
		int kMp = ora * 3600 + perc * 60 + mp;

		System.out.println("Kérem a hívás végének időpontját óra perc másodperc formátumban");
		System.out.print("szóközökkel elválasztva: ");
		ido = sc.nextLine();
		daraboltIdo = ido.split(" ");
		ora = Integer.parseInt(daraboltIdo[0]);
		perc = Integer.parseInt(daraboltIdo[1]);
		mp = Integer.parseInt(daraboltIdo[2]);
		int vMp = ora * 3600 + perc * 60 + mp;

		int hivasIdo = hivasIdoMeghat(kMp, vMp);

		System.out.println("A hívásidő a számlázás szempontjából: " + hivasIdo + " perc. \n");

		sc.close();

		System.out.println("A 3. feladat megoldása");
		System.out.println("Az adatok beolvasása a hivasok.txt fájlból");
		System.out.println("és a hívásidők kiírása a percek.txt fájlba");

		BufferedReader beolvas = new BufferedReader(new FileReader("hivasok.txt"));
```

```
String sor;
String[] telSzam = new String[200];
boolean[] csucsIdo = new boolean[200];
int[] hIdo = new int[200];// hívásidő
String[] hJelleg = new String[200];

int db = 0;

while ((sor = beolvas.readLine()) != null) {
        db++;
        daraboltIdo = sor.split(" ");

        ora = Integer.parseInt(daraboltIdo[0]);
        perc = Integer.parseInt(daraboltIdo[1]);
        mp = Integer.parseInt(daraboltIdo[2]);
        kMp = ora * 3600 + perc * 60 + mp;

        csucsIdo[db - 1] = (ora >= 7) && (ora < 18);

        ora = Integer.parseInt(daraboltIdo[3]);
        perc = Integer.parseInt(daraboltIdo[4]);
        mp = Integer.parseInt(daraboltIdo[5]);
        vMp = ora * 3600 + perc * 60 + mp;

        hIdo[db - 1] = hivasIdoMeghat(kMp, vMp);

        telSzam[db - 1] = beolvas.readLine();
        hJelleg[db - 1] = jellegMeghat(telSzam[db - 1]);
}

beolvas.close();

PrintWriter kiir = new PrintWriter(new FileWriter("percek.txt"));

for (int i = 1; i <= db; i++) {
        kiir.println(hIdo[i - 1] + " " + telSzam[i - 1]);
}

kiir.close();

System.out.println("A fájlkiírás befejeződött. \n");

System.out.println("A 4. feladat megoldása");
System.out.println("Hívások száma csúcsidőben és csúcsidőn kívül");
// A csúcsidő eldöntése már a beolvasáskor megtörtént.

int csucsDb = 0;
int kivulDb = 0;

for (int i = 1; i <= db; i++) {
        if (csucsIdo[i - 1]) {
                csucsDb++;
        } else {
                kivulDb++;
        }
}

System.out.println("Összes hívás: " + db);
System.out.println("Csúcsidőben: " + csucsDb);
System.out.println("Csúcsidőn kívül: " + kivulDb);
System.out.println("");

System.out.println("Az 5. feladat megoldása");
System.out.println("A hívásidő kiszámítása mobil, ill. vezetékes hívásoknál");

// A hívások jellegének eldöntése már a beolvasáskor megtörtént.
```

```
                int mobilIdo = 0;
                int vezetekesIdo = 0;

                for (int i = 1; i <= db; i++) {
                        if (hJelleg[i - 1].equals("mobil")) {
                                mobilIdo += hIdo[i - 1];
                        } else {
                                vezetekesIdo += hIdo[i - 1];
                        }
                }
                System.out.println("Mobil szám: " + mobilIdo + " perc");
                System.out.println("Vezetékes szám: " + vezetekesIdo + " perc");
                System.out.println("");

                System.out.println("A 6. feladat megoldása");
                System.out.println("Hívásdíj csúcsidőben");

                double mobilCsucsTarifa = 69.175;
                double vezetekesCsucsTarifa = 30;
                double csucsDij = 0;

                for (int i = 1; i <= db; i++) {
                        if (csucsIdo[i - 1]) {
                                if (hJelleg[i - 1].equals("mobil")) {
                                        csucsDij += hIdo[i - 1] * mobilCsucsTarifa;
                                } else {
                                        csucsDij += hIdo[i - 1] * vezetekesCsucsTarifa;
                                }
                        }
                }
                System.out.printf("A felhasználónak a csúcsidős hívásokért %.2f", csucsDij);
                System.out.println(" Ft-ot kell fizetnie. \n");
        }

        public static String jellegMeghat(String telSzam) {

                if (telSzam.substring(0, 2).equals("39") || telSzam.substring(0, 2).equals("41")
                                || telSzam.substring(0, 2).equals("79")) {
                        return "mobil";
                } else {
                        return "vezetékes";
                }
        }

        public static int hivasIdoMeghat(int kMp, int vMp) {
                int idoMp = vMp - kMp;
                if (idoMp % 60 == 0) {
                        return idoMp / 60; // Egészosztás, csak az egészrészt adja meg!
                } else {
                        return idoMp / 60 + 1; // A megkezdett perceket felfelé kerekítjük
                }
        }

}
```

```
Az 1. feladat megoldása
Egy billentyűzetről bekért telefonszám jellegének meghatározása
Kérem a telefonszámot: 123456789
A megadott telefonszám: vezetékes

A 2. feladat megoldása
A hívásidő kiszámítása a bekért időpontok alapján
Kérem a hívás kezdetének időpontját óra perc másodperc formátumban
szóközökkel elválasztva: 6 7 8
Kérem a hívás végének időpontját óra perc másodperc formátumban
szóközökkel elválasztva: 6 8 7
A hívásidő a számlázás szempontjából: 1 perc.

A 3. feladat megoldása
Az adatok beolvasása a hivasok.txt fájlból
és a hívásidők kiírása a percek.txt fájlba
A fájlkiírás befejeződött.

A 4. feladat megoldása
Hívások száma csúcsidőben és csúcsidőn kívül
Összes hívás: 85
Csúcsidőben: 63
Csúcsidőn kívül: 22

Az 5. feladat megoldása
A hívásidő kiszámítása mobil, ill. vezetékes hívásoknál
Mobil szám: 53 perc
Vezetékes szám: 293 perc

A 6. feladat megoldása
Hívásdíj csúcsidőben
A felhasználónak a csúcsidős hívásokért 8807,93 Ft-ot kell fizetnie.
```

A percek.txt szöveges fájl

2 392712621	1 716481028	2 614761384
5 442407028	4 796823702	5 682503335
5 712676212	1 394301623	2 646830328
8 297241726	2 361206275	2 713544421
9 565866886	5 716573357	3 874533786
9 166566516	2 431546527	6 528208481
8 865826206	2 394535174	2 173583677
5 414586306	2 138100078	9 622713308
6 716118757	1 281685640	5 391505271
3 565866886	6 565866886	9 424408282
5 231215421	3 113207278	4 783426333
3 696500077	3 281685640	8 385641234
3 583643771	4 375480805	5 472184487
2 392516252	3 565758448	9 432752572
1 762256824	2 714346720	6 134771866
1 716646345	3 716573357	5 718243750
2 631431046	3 715704877	3 284424535
4 398768266	2 881157107	3 661836544
3 631283182	6 473177057	7 565866886
3 392746060	1 714346720	9 822587003
1 824238334	6 412410573	5 268587300
2 397618418	5 478630720	5 734468211
4 267505842	5 867323353	2 263134032
2 281685640	1 767445223	5 712272350
5 212763810	1 688861311	10 713563064
3 281685640	4 477337448	2 682524752
5 392746060	6 418861311	2 484566325
3 716573357	4 221623208	
6 392746060	5 165555618	

2006 május: Fehérje

A fehérjék óriás molekulák, amelyeknek egy része az élő szervezetekben végbemenő folyamatokat katalizálják. Egy-egy fehérje aminosavak százaiból épül fel, melyek láncszerűen kapcsolódnak egymáshoz. A természetben a fehérjék fajtája több millió. Minden fehérje húszféle aminosav különböző mennyiségű és sorrendű összekapcsolódásával épül fel.

Az alábbi táblázat tartalmazza az aminosavak legfontosabb adatait, a megnevezéseket és az őket alkotó atomok számát (az aminosavak mindegyike tartalmaz szenet, hidrogént, oxigént és nitrogént, néhányban kén is van):

Neve	Rövidítés	Betűjele	C	H	O	N	S
Glicin	Gly	G	2	5	2	1	0
Alanin	Ala	A	3	7	2	1	0
Arginin	Arg	R	6	14	2	4	0
Fenilalanin	Phe	F	9	11	2	1	0
Cisztein	Cys	C	3	7	2	1	1
Triptofán	Trp	W	11	12	2	2	0
Valin	Val	V	5	11	2	1	0
Leucin	Leu	L	6	13	2	1	0
Izoleucin	Ile	I	6	13	2	1	0
Metionin	Met	M	5	11	2	1	1
Prolin	Pro	P	5	9	2	1	0
Szerin	Ser	S	3	7	3	1	0
Treonin	Thr	T	4	9	3	1	0
Aszparagin	Asn	N	4	8	3	2	0
Glutamin	Gln	Q	5	10	3	2	0
Tirozin	Tyr	Y	9	11	3	1	0
Hisztidin	His	H	6	9	2	3	0
Lizin	Lys	K	6	14	2	2	0
Aszparaginsav	Asp	D	4	7	4	1	0
Glutaminsav	Glu	E	5	9	4	1	0

Készítsen programot *feherje* néven, ami megoldja a következő feladatokat! Ügyeljen arra, hogy a program forráskódját a megadott helyre mentse!

1. Töltse be az *aminosav.txt* fájlból az aminosavak adatait! A fájlban minden adat külön sorban található, a fájl az aminosavak nevét nem tartalmazza. Ha az adatbetöltés nem sikerül, vegye fel a fenti táblázat alapján állandóként az első öt adatsort, és azzal dolgozzon!

Az első néhány adat:

Gly
G
2
5
2
1
0
Ala
A
3
7
2
1
0
…

2. Határozza meg az aminosavak relatív molekulatömegét, ha a szén atomtömege 12, a hidrogéné 1, az oxigéné 16, a nitrogéné 14 és a kén atomtömege 32! Például a Glicin esetén a relatív molekulatömeg 2·12 + 5·1 + 2·16 + 1·14 + 0·32 = 75.

A következő feladatok eredményeit írja képernyőre, illetve az *eredmeny.txt* fájlba! A kiírást a feladat sorszámának feltüntetésével kezdje (például: 4. feladat)!

3. Rendezze növekvő sorrendbe az aminosavakat a relatív molekulatömeg szerint! Írja ki a képernyőre és az *eredmeny.txt* fájlba az aminosavak hárombetűs azonosítóját és a molekulatömeget! Az azonosítót és hozzátartozó molekulatömeget egy sorba, szóközzel elválasztva írja ki!

4. A *bsa.txt* a *BSA* nevű fehérje aminosav sorrendjét tartalmazza – egybetűs jelöléssel. (A fehérjelánc legfeljebb 1000 aminosavat tartalmaz.) Határozza meg a fehérje összegképletét (azaz a C, H, O, N és S számát)! A meghatározásánál vegye figyelembe, hogy az aminosavak összekapcsolódása során minden kapcsolat létrejöttekor egy vízmolekula (H_2O) lép ki! Az összegképletet a képernyőre és az *eredmeny.txt* fájlba az alábbi formában írja ki:
Például: C 16321 H 34324 O 4234 N 8210 S 2231

(Amennyiben a bsa.txt beolvasása sikertelen, helyette tárolja a G,A,R,F,C betűjeleket tízszer egymás után és a feladatokat erre a „láncra” oldja meg!)

5. A fehérjék szekvencia szerkezetét hasításos eljárással határozzák meg. Egyes enzimek bizonyos aminosavak után kettéhasítják a fehérjemolekulát. Például a Kimotripszin enzim a Tirozin (Y), Fenilalanin (W) és a Triptofán (F) után hasít.
Határozza meg, és írja ki képernyőre a Kimotripszin enzimmel széthasított BSA lánc leghosszabb darabjának hosszát és az eredeti láncban elfoglalt helyét (első és utolsó aminosavának sorszámát)! A kiíráskor nevezze meg a kiírt adatot, például: „kezdet helye:”!

6. Egy másik enzim (a Factor XI) az Arginin (R) után hasít, de csak akkor, ha Alinin (A) vagy Valin (V) követi. Határozza meg, hogy a hasítás során keletkező első fehérjelánc részletben hány Cisztein (C) található! A választ teljes mondatba illesztve írja ki a képernyőre!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * Fehérje
 *
 * @author Klemand
 */

public class EmeltInfo2006maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása az aminosav.txt fájlból \n");

		BufferedReader beolvas = new BufferedReader(new FileReader("aminosav.txt"));

		String sor;
		String[] rov = new String[20];
		String[] jel = new String[20];
		int[][] tul = new int[20][6];
		//C12, H1, O16, N14, S32 darabszáma és a móltömeg

		int db = 0;
		int i, j;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			rov[db - 1] = sor;
			jel[db - 1] = beolvas.readLine();
			for(j = 1; j <= 5; j++) {
				tul[db - 1][j-1] = Integer.parseInt(beolvas.readLine());
			}
		}
		beolvas.close();

		System.out.println("A 2. feladat megoldása");
		System.out.println("A relatív molekulatömegek meghatározása \n");
		int[] atomSzam = {12, 1, 16, 14, 32};
		for (i = 1; i <= db; i++) {
			tul[i - 1][5] = 0;
			for(j = 1; j<= 5; j++) {
				tul[i - 1][5] += atomSzam[j - 1] * tul[i - 1][j - 1];
			}
		}
```

```
System.out.println("A 3. feladat megoldása");
System.out.println("Rendezés a relatív molekulatömegek szerint");
// buborékos rendezés

String StringAsztal;
int[] intTombAsztal = new int[6];

for (i = db - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (tul[j - 1][5] > tul[j][5]) {

                        StringAsztal = rov[j - 1];
                        rov[j - 1] = rov[j];
                        rov[j] = StringAsztal;

                        StringAsztal = jel[j - 1];
                        jel[j - 1] = jel[j];
                        jel[j] = StringAsztal;

                        intTombAsztal = tul[j - 1];
                        tul[j - 1] = tul[j];
                        tul[j] = intTombAsztal;

                }
        }
}

System.out.println("Kiírás az eredmeny.txt fájlba");

PrintWriter kiir = new PrintWriter(new FileWriter("eredmeny.txt"));

kiir.println("A 3. feladat megoldása");

for (i = 1; i <= db; i++) {
        kiir.println(rov[i - 1] + " " + tul[i - 1][5]);
}

kiir.println("");
System.out.println("A fájlkiírás befejeződött. \n");

System.out.println("A 4. feladat megoldása");
System.out.println("A bsa.txt fájl beolvasása");

String[] bsa = new String[1000];
int bsaDb = 0;

beolvas = new BufferedReader(new FileReader("bsa.txt"));

while ((sor = beolvas.readLine()) != null) {
        bsaDb++;
        bsa[bsaDb - 1] = sor;
}
beolvas.close();
```

```
System.out.print("A BSA nevű fehérje összegképlete: ");
int[] elemDb = new int[5];
int k = 0;
for(k = 1; k <= 5; k++) {
        elemDb[k - 1] = 0;
}

for (i = 1; i <= bsaDb; i++) {
        // Az aktuális aminosav kiválasztása
        j = 1;
        while (!jel[j - 1].equals(bsa[i - 1])) {
                j++;
        }
        for(k = 1; k <= 5; k++) {
                elemDb[k - 1] += tul[j - 1][k - 1];
        };
}
// A vízmolekulák kilépésének beszámítása
elemDb[1] -= 2 * (bsaDb - 1);
elemDb[2] -= bsaDb - 1;

String[] atomJel = {"C", "H", "O", "N", "S"};
for(k = 1; k <= 5; k++) {
        System.out.print(atomJel[k - 1] + " " + elemDb[k - 1] + " ");
};
System.out.println();

System.out.println("Kiírás folytatása az eredmeny.txt fájlba");
kiir.println("A 4. feladat megoldása");
kiir.print("A BSA nevű fehérje összegképlete: ");

for(k = 1; k <= 5; k++) {
        kiir.print(atomJel[k - 1] + " " + elemDb[k - 1] + " ");
};
kiir.close();
System.out.println("A fájlkiírás befejeződött. \n");

System.out.println("Az 5. feladat megoldása");

int aktTkezdet = -1; // irreális kezdőérték
int aktThossz = 0;
int maxTkezdet = -1; // irreális kezdőérték
int maxThossz = 0;

for (i = 1; i <= bsaDb; i++) {
        // A hasítás a megadott elemek után történik!
        aktThossz++;
        if (aktTkezdet == -1) {
                aktTkezdet = i;
        }
        if (aktThossz > maxThossz) {
                maxTkezdet = aktTkezdet;
                maxThossz = aktThossz;
        }
        if (hasitKimotripszin(bsa[i - 1])) {
                aktTkezdet = -1;
                aktThossz = 0;
        }
}

System.out.println("A Kimotripszin enzimmel széthasított BSA lánc leghosszabb darabja:");
System.out.println("Kezdete: " + maxTkezdet + ". elem: " + bsa[maxTkezdet - 1]);
System.out.print("Vége: " + (maxTkezdet + maxThossz - 1));
System.out.println(". elem: " + bsa[maxTkezdet + maxThossz - 2]);
System.out.println("Hossza: " + maxThossz);
System.out.println("A leghosszabb láncdarab előtti hasító aminosav: " + bsa[maxTkezdet - 2]);
System.out.println("");
```

```
                System.out.println("A 6. feladat megoldása");
                System.out.println("A Ciszteinek számának meghatározása a a Factor XI első hasítási láncában");
                int ciszteinDb = 0;
                i = 1;
                while (i < bsaDb && !(hasitFactorXI(bsa[i - 1], bsa[i]))) {
                        if (bsa[i - 1].equals("C")) {
                                ciszteinDb++;
                        }
                        i++;
                }
                System.out.println("Az első hasítási láncban " + ciszteinDb + " Cisztein található. \n");
        }

        public static boolean hasitKimotripszin(String a) {
                return (a.equals("Y") || a.equals("W") || a.equals("F"));
        }

        public static boolean hasitFactorXI(String a, String b) {
                return (a.equals("R") && (b.equals("A") || b.equals("V")));
        }

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása az aminosav.txt fájlból

A 2. feladat megoldása
A relatív molekulatömegek meghatározása

A 3. feladat megoldása
Rendezés a relatív molekulatömegek szerint
Kiírás az eredmeny.txt fájlba
A fájlkiírás befejeződött.

A 4. feladat megoldása
A bsa.txt fájl beolvasása
A BSA nevű fehérje összegképlete: C 2923 H 4594 O 895 N 778 S 39
Kiírás folytatása az eredmeny.txt fájlba
A fájlkiírás befejeződött.

Az 5. feladat megoldása
A Kimotripszin enzimmel széthasított BSA lánc leghosszabb darabja:
Kezdete: 261. elem: I
Vége: 306. elem: F
Hossza: 46
A leghosszabb láncdarab előtti hasító aminosav: Y

A 6. feladat megoldása
A Ciszteinek számának meghatározása a a Factor XI első hasítási láncában
Az első hasítási láncban 12 Cisztein található.
```

Az eredmeny.txt szovegfájl

```
A 3. feladat megoldása
Gly 75
Ala 89
Ser 105
Pro 115
Val 117
Thr 119
Cys 121
Leu 131
Ile 131
Asn 132
Asp 133
Gln 146
Lys 146
Glu 147
Met 149
His 155
Phe 165
Arg 174
Tyr 181
Trp 204

A 4. feladat megoldása
A BSA nevű fehérje összegképlete: C 2923 H 4594 O 895 N 778 S 39
```

2006 október: Zenei adók

A rádióhallgatás ma már egyre inkább zene vagy hírek hallgatására korlátozódik. Ez a feladat három, folyamatosan zenét sugárzó adóról szól, azok egyetlen napi műsorát feldolgozva. A reklám elkerülése érdekében az adókat nevük helyett egyetlen számmal azonosítottuk.

A *musor.txt* állomány első sorában az olvasható, hogy hány zeneszám (z<=1000) szólt aznap a rádiókban, majd ezt z darab sor követi. Minden sor négy, egymástól egyetlen szóközzel elválasztott adatot tartalmaz: a rádió sorszámát, amit a szám hossza követ két egész szám (perc és másodperc) formában, majd a játszott szám azonosítója szerepel, ami a szám előadójából és címéből áll. A rádió sorszáma az 1, 2, 3 számok egyike. Az adás minden adón 0 óra 0 perckor kezdődik. Egyik szám sem hosszabb 30 percnél, tehát a perc értéke legfeljebb 30, a másodperc pedig legfeljebb 59 lehet. A szám azonosítója legfeljebb 50 karakter hosszú, benne legfeljebb egy kettőspont szerepel, ami az előadó és a cím között található. A számok az elhangzás sorrendjében szerepelnek az állományban, tehát a később kezdődő szám későbbi sorban található. Az állományban minden zeneszám legfeljebb egyszer szerepel.

Például:

```
677
1 5 3 Deep Purple:Bad Attitude
2 3 36 Eric Clapton:Terraplane Blues
3 2 46 Eric Clapton:Crazy Country Hop
3 3 25 Omega:Ablakok
…
```

Készítsen programot *zene* néven, amely az alábbi kérdésekre válaszol!
Ügyeljen arra, hogy a program forráskódját a megadott helyre mentse!

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: 3. feladat:). Ha a billentyűzetről olvas be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár.

Az adatszerkezet készítése során vegye figyelembe az Ön által használt programozási környezetben az adatok tárfoglalási igényét!

1. Olvassa be a *musor.txt* állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, akkor a forrás első 10 sorának adatait jegyezze be a programba, s úgy oldja meg a következő feladatokat!
2. Írja a képernyőre, hogy melyik csatornán hány számot lehetett meghallgatni!
3. Adja meg, mennyi idő telt el az első Eric Clapton szám kezdete és az utolsó Eric Clapton szám vége között az 1. adón! Az eredményt *óra:perc:másodperc* formában írja a képernyőre!
4. Amikor az „Omega:Legenda” című száma elkezdődött, Eszter rögtön csatornát váltott. Írja a képernyőre, hogy a szám melyik adón volt hallható, és azt, hogy a másik két adón milyen számok szóltak ekkor. Mivel a számok a kezdés időpontja szerint növekvő sorrendben vannak, így a másik két adón már elkezdődött a számok lejátszása. Feltételezheti, hogy a másik két adón volt még adás.

5. Az egyik rádióműsorban sms-ben, telefonon, de akár képeslapon is kérhető szám. Ám a sokszor csak odafirkált kéréseket olykor nehéz kibetűzni. Előfordul, hogy csak ennyi olvasható: „gaoaf”, tehát ezek a betűk biztosan szerepelnek, mégpedig pontosan ebben a sorrendben. Annyi biztos, hogy először a szerző neve szerepel, majd utána a szám címe. Olvassa be a billentyűzetről a felismert karaktereket, majd írja a `keres.txt` állományba azokat a számokat, amelyek ennek a feltételnek megfelelnek. Az állomány első sorába a beolvasott karaktersorozat, majd utána soronként egy zeneszám azonosítója kerüljön! A feladat megoldása során ne különböztesse meg a kis- és a nagybetűket!

6. Az 1. adón változik a műsor szerkezete: minden számot egy rövid, egyperces bevezető előz majd meg, és műsorkezdéstől minden egész órakor 3 perces híreket mondanak. Természetesen minden szám egy részletben hangzik el továbbra is, közvetlenül a bevezető perc után. Így ha egy szám nem fejeződne be a hírekig, el sem kezdik, az üres időt a műsorvezető tölti ki. Írja a képernyőre *óra:perc:másodperc* formában, hogy mikor lenne vége az adásnak az új műsorszerkezetben!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Zenei adók
 *
 * @author Klemand
 */

public class EmeltInfo2006okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása az musor.txt fájlból");

		BufferedReader beolvas = new BufferedReader(new FileReader("musor.txt"));

		String sor;

		sor = beolvas.readLine();

		int n = Integer.parseInt(sor.trim());
		// Egy szóköz szerepel a szám után a sor végén, le kell vágni!

		int[] ado = new int[n];
		int[] perc = new int[n];
		int[] mp = new int[n];
		String[] azonosito = new String[n];

		for (int i = 1; i <= n; i++) {
			sor = beolvas.readLine();
			/*
			 * Mivel az azonosítóban is vannak szóközök, nekem kell feldarabolnom!
			 */

			ado[i - 1] = Integer.parseInt(sor.substring(0, sor.indexOf(" ")));
			sor = sor.substring(sor.indexOf(" ") + 1); // kezdőindextől végig
			perc[i - 1] = Integer.parseInt(sor.substring(0, sor.indexOf(" ")));
			sor = sor.substring(sor.indexOf(" ") + 1);
			mp[i - 1] = Integer.parseInt(sor.substring(0, sor.indexOf(" ")));
			azonosito[i - 1] = sor.substring(sor.indexOf(" ") + 1);
		}

		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Az egyes csatornákon hallható dalok száma:");

		int db1 = 0;
		int db2 = 0;
		int db3 = 0;
		int i;

		for (i = 1; i <= n; i++) {

			switch (ado[i - 1]) {
			case 1:
				db1++;
				break;
```

```
        case 2:
                db2++;
                break;
        default:
                db3++;
        }
}

System.out.println("1. adó: " + db1);
System.out.println("2. adó: " + db2);
System.out.println("3. adó: " + db3);
System.out.println("");

System.out.println("A 3. feladat megoldása");
System.out.println("Az első Eric Clapton dal kezdete és az utolsó Eric Clapton dal vége");
System.out.print("között eltelt idő az 1. adón: ");

// A dalok kezdési időpontjának meghatározása a nap kezdete óta másodpercben

int[] kezdet = new int[n];
int akt1 = 0;
int akt2 = 0;
int akt3 = 0;

for (i = 1; i <= n; i++) {

        switch (ado[i - 1]) {
        case 1:
                kezdet[i - 1] = akt1; // aktuális időpont az 1. adón
                akt1 = akt1 + perc[i - 1] * 60 + mp[i - 1];
                // a szám vége lesz a következő aktuális időpont az 1. adón
                break;
        case 2:
                kezdet[i - 1] = akt2;
                akt2 = akt2 + perc[i - 1] * 60 + mp[i - 1];
                break;
        default:
                kezdet[i - 1] = akt3;
                akt3 = akt3 + perc[i - 1] * 60 + mp[i - 1];
        }
}

int elsoKezd = -1; // irreális kezdőértékek
int utolsoVeg = -1;
i = 1;
while (i <= n && !EC1(ado[i - 1], azonosito[i - 1])) {
        i++;
}
if (i <= n) {
        elsoKezd = kezdet[i - 1];
        utolsoVeg = kezdet[i - 1] + perc[i - 1] * 60 + mp[i - 1];
        int j;
        for (j = i + 1; j <= n; j++) {
                if (EC1(ado[j - 1], azonosito[j - 1])) {
                        utolsoVeg = kezdet[j - 1] + perc[j - 1] * 60 + mp[j - 1];
                }
        }
        System.out.println(idoKonvertalas(utolsoVeg - elsoKezd));
        System.out.print("(Az első kezdete: " + idoKonvertalas(elsoKezd));
        System.out.println(", az utolsó vége: " + idoKonvertalas(utolsoVeg) + ")\n");
} else {
        System.out.println("Az 1. adón a nap folyamán nem játszottak Eric Clapton számot.\n");
}
```

```
System.out.println("A 4. feladat megoldása tetszőleges műsorszámra");
System.out.println("Add meg egy műsorszám azonosítóját! Pl. Omega:Legenda ");

Scanner sc = new Scanner(System.in);
String musorSzam = sc.nextLine();

System.out.println("Melyik adón, és mikor játszották a megadott számot?");

int melyik = -1;
int mikor = -1;
int masik1 = -1;
int masik2 = -1;
i = 1;

while (i <= n && !azonosito[i - 1].equals(musorSzam)) {
        i++;
}

if (i <= n) {
        melyik = ado[i - 1];
        mikor = kezdet[i - 1];

        System.out.println("Műsorszám: " + musorSzam);
        System.out.println("Adó: " + melyik);
        System.out.println("Kezdési időpont: " + idoKonvertalas(mikor));
        System.out.println("");

        System.out.println("Mit játszottak ekkor a másik két adón?");

        int j = i - 1;
        while (j >= 1 && ado[j - 1] == melyik) {
                j--;
        }

        if (j >= 1) {
                masik1 = ado[j - 1];
                if (kezdet[j - 1] + perc[j - 1] * 60 + mp[j - 1] >= mikor) {
                        // Ezt feltételezhetnénk mindkét adóra, de a megadott példában nem teljesül
                        System.out.println("Adó: " + masik1);
                        System.out.println("Játszott szám: " + azonosito[j - 1]);
                        System.out.println("Kezdési időpont: " + idoKonvertalas(kezdet[j - 1]));
                } else {
                        System.out.println("A(z) " + masik1 + " adón már befejeződött a műsor.");
                }
        }

        j--;
        while (j >= 1 && (ado[j - 1] == melyik || ado[j - 1] == masik1)) {
                j--;
        }

        if (j >= 1) {
                masik2 = ado[j - 1];
                if (kezdet[j - 1] + perc[j - 1] * 60 + mp[j - 1] >= mikor) {
                        System.out.println("Adó: " + masik2);
                        System.out.println("Játszott szám: " + azonosito[j - 1]);
                        System.out.println("Kezdési időpont: " + idoKonvertalas(kezdet[j - 1]));
                } else {
                        System.out.println("A(z) " + masik2 + " adón már befejeződött a műsor.");
                }
        }
} else {
        System.out.println("A megadott számot egyik adón sem játszották.");
}
System.out.println();
```

```
        System.out.println("Az 5. feladat megoldása");
        System.out.println("Adj meg egy karaktersorozatot! Pl. gaoaf ");
        System.out.println("A program kiírja a keres.txt fájlba,");
        System.out.println("hogy mely számokban szerepelnek a megadott karakterek");
        System.out.println("a megadott sorrendben, ha a kis- és a nagybetűk között nem teszünk különbséget.");

        String foszlany = sc.nextLine();
        sc.close();
        System.out.println();

        PrintWriter kiir = new PrintWriter(new FileWriter("keres.txt"));

        kiir.println(foszlany);
        foszlany = foszlany.toLowerCase();

        for (i = 1; i <= n; i++) {
                if (benneVan(foszlany, azonosito[i - 1])) {
                        kiir.println(azonosito[i - 1]);
                }
        }

        kiir.close();
        System.out.println("A fájlkiírás befejeződött. \n");

        System.out.println("A 6. feladat megoldása");
        System.out.println("Új műsorstruktúra az 1. adón ");

        int kezdes = 0;
        int vege = 0;
        int ora = 0;

        for (i = 1; i <= n; i++) {

                if (ado[i - 1] == 1) {
                        kezdes = vege;
                        vege = kezdes + 60 + perc[i - 1] * 60 + mp[i - 1];
                        if (vege > (ora + 1) * 3600) {
                                // ha a számnak nem lenne vége a hírekig
                                ora++;
                                kezdes = ora * 3600 + 180;
                                // akkor csak a hírek után kezdődik
                                vege = kezdes + 60 + perc[i - 1] * 60 + mp[i - 1];
                        }
                }
        }

        String befejezes = idoKonvertalas(vege);
        System.out.println("Az új műsorszerkezetben az 1. adó műsorának befejezése: " + befejezes + "\n");
}
```

```
	public static boolean EC1(int ado, String azon) {
		return ado == 1 && azon.split(":")[0].equals("Eric Clapton");
	}

	public static String idoKonvertalas(int ido) {
		int ora, perc, mp;
		String konv = "";
		mp = ido % 60;
		perc = ido / 60;
		ora = perc / 60;
		perc = perc % 60;

		if (ora < 10) {
			konv += "0";
		}
		konv += ora + ":";

		if (perc < 10) {
			konv += "0";
		}
		konv += perc + ":";

		if (mp < 10) {
			konv += "0";
		}
		konv += mp;

		return konv;
		// visszaadott függvényérték
	}

	public static boolean benneVan(String foszlany, String azonosito) {
		azonosito = azonosito.toLowerCase();
		int m = foszlany.length();
		String betu;
		int i = 1;
		int k = -1;
		boolean talalt = true;

		while (i <= m && talalt) {
			betu = foszlany.substring(i - 1, i);
			k = azonosito.indexOf(betu);
			if (k > -1) {
				azonosito = azonosito.substring(k + 1);
				i++;
			} else {
				talalt = false;
			}
		}
		return talalt;
	}

}
```

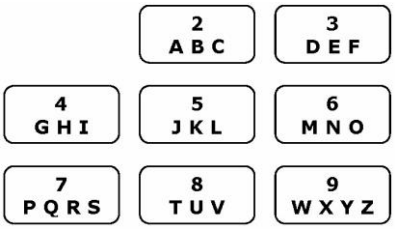
```
Az 1. feladat megoldása
Az adatok beolvasása az musor.txt fájlból
A beolvasás megtörtént.

A 2. feladat megoldása
Az egyes csatornákon hallható dalok száma:
1. adó: 232
2. adó: 215
3. adó: 230

A 3. feladat megoldása
Az első Eric Clapton dal kezdete és az utolsó Eric Clapton dal vége
között eltelt idő az 1. adón: 17:51:12
(Az első kezdete: 00:05:03, az utolsó vége: 17:56:15)

A 4. feladat megoldása tetszőleges műsorszámra
Add meg egy műsorszám azonosítóját! Pl. Omega:Legenda
Omega:Legenda
Melyik adón, és mikor játszották a megadott számot?
Műsorszám: Omega:Legenda
Adó: 3
Kezdési időpont: 16:45:08

Mit játszottak ekkor a másik két adón?
Adó: 1
Játszott szám: Eric Clapton:Grand Illusion
Kezdési időpont: 16:41:30
A(z) 2 adón már befejeződött a műsor.

Az 5. feladat megoldása
Adj meg egy karaktersorozatot! Pl. gaoaf
A program kiírja a keres.txt fájlba,
hogy mely számokban szerepelnek a megadott karakterek
a megadott sorrendben, ha a kis- és a nagybetűk között nem teszünk különbséget.
gaoaf

A fájlkiírás befejeződött.

A 6. feladat megoldása
Új műsorstruktúra az 1. adón
Az új műsorszerkezetben az 1. adó műsorának befejezése: 23:44:41
```

A keres.txt szöveges fájl

```
gaoaf
Omega:Rozsafak
Omega:Trombitas Fredi
Omega:Ballada a fegyverkovacs fiarol
```

2007 május: SMS szavak

Napjainkban a kommunikáció egy elterjedt formája az SMS-küldés. Az SMS-küldésre alkalmas telefonok prediktív szövegbevitellel segítik az üzenetek megírását. Ennek használatakor a szavakat úgy tudjuk beírni, hogy a telefon számbillentyűjén található betűknek megfelelő számokat kell beírnunk. A számok és betűk megfeleltetését az alábbi táblázat mutatja:



Ha meg szeretnénk jeleníteni az „*ablak*” szót, akkor a *22525* kódot kell beírnunk. A telefon a tárolt szótára alapján a kódhoz kikeresi a megfelelő szót. Ha több szóhoz is azonos kód tartozik, akkor a kódhoz tartozó összes szót felkínálja választásra. Egy ilyen szógyűjteményt talál a `szavak.txt` fájlban.
A fájlról a következőket tudjuk:

- Legfeljebb 600 szó található benne.
- Minden szó külön sorban található.
- A szavak hossza maximum 15 karakter.
- A szavak mindegyike csak az angol ábécé kisbetűit tartalmazza.
- Minden szó legfeljebb egyszer szerepel.

Írjon `sms` néven programot, ami a szógyűjtemény felhasználásával megoldja az alábbi feladatokat!

1. Kérjen be a felhasználótól egy betűt, és adja meg, hogy milyen kód (szám) tartozik hozzá! Az eredményt írassa a képernyőre!

2. Kérjen be a felhasználótól egy szót, és határozza meg, hogy milyen számsorral lehet ezt a telefonba bevinni! Az eredményt írassa a képernyőre!

3. Olvassa be a `szavak.txt` fájlból a szavakat, és a továbbiakban azokkal dolgozzon! Ha nem tudja az állományból beolvasni az adatokat, akkor az állományban található „b” kezdőbetűs szavakat gépelje be a programba, és azokkal oldja meg a feladatokat!

4. Határozza meg és írassa a képernyőre, hogy melyik a leghosszabb tárolt szó! Amennyiben több azonos hosszúságú van, elegendő csak az egyiket megjeleníteni.
   Adja meg ennek a szónak a hosszát is!

5. Határozza meg és írassa a képernyőre, hogy hány rövid szó található a fájlban! Rövid szónak tekintjük a legfeljebb 5 karakterből álló szavakat.

6. Írassa a `kodok.txt` állományba a `szavak.txt` fájlban található szavaknak megfelelő számkódokat! Minden szónak feleljen meg egy számkód, és minden számkód külön sorba kerüljön!

7. Kérjen be a felhasználótól egy számsort, és határozza meg, hogy melyik szó tartozhat hozzá! Amennyiben több szó is megfelelő, akkor mindegyiket írassa ki! (Teszteléshez használhatja például a *225* számsort, mivel ehhez egynél több szó tartozik a szógyűjteményben.)

8. Határozza meg, hogy a szógyűjteményben mely kódokhoz tartozik több szó is! Írassa ki a képernyőre ezeket a szavakat a kódjukkal együtt egymás mellé az alábbi mintának megfelelően (a szavak sorrendje ettől eltérhet):

```
baj : 225; bal : 225; arc : 272; apa : 272; eb : 32; fa : 32; dal : 325;
fal : 325; eltesz : 358379; elvesz : 358379; fojt : 3658; folt : 3658;
...
```

9. Határozza meg, hogy melyik kódnak megfelelő szóból van a legtöbb! Írassa ki a képernyőre a kódot, és a kódhoz tartozó összes tárolt szót! Ha több kódhoz is azonos számú szó tartozik, akkor elegendő ezen kódok közül csak az egyikkel foglalkozni.

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * SMS szavak
 *
 * @author Klemand
 */

public class EmeltInfo2007maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat  megoldása");
		System.out.println("A billentyűzetről beolvasott betű kódjának meghatározása");

		Scanner sc = new Scanner(System.in);

		System.out.print("Kérek egy betűt: ");
		String betu = sc.nextLine();
		betu = betu.toLowerCase();

		String kod = kodol(betu);
		System.out.println("A(z) " + betu + " kódja: " + kod + "\n");

		System.out.println("A 2. feladat  megoldása");
		System.out.println("Egy szó kódjának meghatározása\n");

		System.out.print("Kérek egy (ékezetes betűk nélküli) szót: ");

		String szo = sc.nextLine();
		szo = szo.toLowerCase();

		kod = "";
		int i;

		for (i = 1; i <= szo.length(); i++) {
			kod += kodol(szo.substring(i - 1, i));
		}

		System.out.println("A(z) " + szo + " kódja: " + kod + "\n");

		System.out.println("A 3. feladat  megoldása");
		System.out.println("Az adatok beolvasása a szavak.txt fájlból");

		BufferedReader beolvas = new BufferedReader(new FileReader("szavak.txt"));

		String[] szavak = new String[600];
		String sor;
		int db = 0;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			szavak[db - 1] = sor;
		}

		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");
```

```
System.out.println("A 4. feladat  megoldása");
System.out.print("A leghosszabb szó meghatározása");

int max = 1;

for (i = 2; i <= db; i++) {
        if (szavak[i - 1].length() > szavak[max - 1].length()) {
                max = i;
        }
}

System.out.print("A szó: " + szavak[max - 1]);
System.out.println(", hossza: " + szavak[max - 1].length() + "\n");

System.out.println("Az 5. feladat  megoldása");
System.out.print("A rövid (legfeljebb 5 karakterből álló) szavak száma: ");

int rdb = 0;

for (i = 1; i <= db; i++) {
        if (szavak[i - 1].length() <= 5) {
                rdb++;
        }
}

System.out.println(rdb + "\n");

System.out.println("A 6. feladat  megoldása");
System.out.println("A szavak kódjainak kiírása a kodok.txt fájlba ");

PrintWriter kiir = new PrintWriter(new FileWriter("kodok.txt"));

String[] kodok = new String[600];
int j;

for (i = 1; i <= db; i++) {
        szo = szavak[i - 1];
        kodok[i - 1] = "";

        for (j = 1; j <= szo.length(); j++) {
                kodok[i - 1] += kodol(szo.substring(j - 1, j));
        }
        kiir.println(kodok[i - 1]);
}

kiir.close();
System.out.println("A fájlkiírás befejeződött. \n");

System.out.println("A 7. feladat  megoldása");
System.out.println("Egy billentyűzetről bekért kódhoz tartozó szavak meghatározása\n ");

System.out.print("Kérek egy számsort: ");
kod = sc.nextLine();

for (i = 1; i <= db; i++) {
        if (kodok[i - 1].equals(kod)) {
                System.out.print(szavak[i - 1] + " ");
        }
}

sc.close();
System.out.println("\n");
```

```
        System.out.println("A 8. feladat  megoldása");
        System.out.println("A több szóhoz rendelt kódok meghatározása");

        // A kódok gyakorisági táblázatának elkészítése
        String[] kodLista = new String[600];
        int[] gyakorisag = new int[600];
        String aktKod;
        int kdb = 0;

        for (i = 1; i <= db; i++) {
                j = 1;
                /*
                 * Az aktuális kód keresése a kódlistában. Ha szerepel benne, növeljük a
                 * darabszámát, ha nem, akkor hozzávesszük a kódlistához.
                 */
                aktKod = kodok[i - 1];
                while ((j <= kdb) && !(aktKod.equals(kodLista[j - 1]))) {
                        j++;
                }

                if (j <= kdb) {
                        gyakorisag[j - 1]++;
                } else {
                        kdb++;
                        kodLista[kdb - 1] = aktKod;
                        gyakorisag[kdb - 1] = 1;
                }
        }

        int tdb = 0; // a többszörös kódok darabszáma a tördeléshez
        for (i = 1; i <= kdb; i++) {
                if (gyakorisag[i - 1] > 1) {
                        aktKod = kodLista[i - 1];
                        for (j = 1; j <= db; j++) {
                                if (kodok[j - 1].equals(aktKod)) {
                                        tdb++;
                                        System.out.print(szavak[j - 1] + " : " + aktKod + "; ");
                                        if (tdb % 7 == 0) {
                                                System.out.println();
                                        }
                                }
                        }
                }
        }
        System.out.println("\n");

        System.out.println("A 9. feladat  megoldása");
        System.out.print("A leggazdagabb kód: ");

        int leggazdagabb = 1;
        for (i = 2; i <= kdb; i++) {
                if (gyakorisag[i - 1] > gyakorisag[leggazdagabb - 1]) {
                        leggazdagabb = i;
                }
        }

        aktKod = kodLista[leggazdagabb - 1];
        System.out.println(aktKod);
        System.out.print(gyakorisag[leggazdagabb - 1] + " szó tartozik hozzá: ");

        for (i = 1; i <= db; i++) {
                if (kodok[i - 1].equals(aktKod)) {
                        System.out.print(szavak[i - 1] + " ");
                }
        }
        System.out.println("\n");
}
```

```
        public static String kodol(String betu) {

                if ("abc".contains(betu)) {
                        return "2";
                } else if ("def".contains(betu)) {
                        return "3";
                } else if ("ghi".contains(betu)) {
                        return "4";
                } else if ("jkl".contains(betu)) {
                        return "5";
                } else if ("mno".contains(betu)) {
                        return "6";
                } else if ("pqrs".contains(betu)) {
                        return "7";
                } else if ("tuv".contains(betu)) {
                        return "8";
                } else if ("wxyz".contains(betu)) {
                        return "9";
                } else {
                        return "";
                }
        }

}
```

```
Az 1. feladat  megoldása
A billentyűzetről beolvasott betű kódjának meghatározása
Kérek egy betűt: g
A(z) g kódja: 4

A 2. feladat  megoldása
Egy szó kódjának meghatározása

Kérek egy (ékezetes betűk nélküli) szót: program
A(z) program kódja: 7764726

A 3. feladat  megoldása
Az adatok beolvasása a szavak.txt fájlból
A beolvasás megtörtént.

A 4. feladat  megoldása
A leghosszabb szó meghatározásaA szó: megfeledkezik, hossza: 13

Az 5. feladat  megoldása
A rövid (legfeljebb 5 karakterből álló) szavak száma: 279

A 6. feladat  megoldása
A szavak kódjainak kiírása a kodok.txt fájlba
A fájlkiírás befejeződött.

A 7. feladat  megoldása
Egy billentyűzetről bekért kódhoz tartozó szavak meghatározása

Kérek egy számsort: 3658
fojt folt

A 8. feladat  megoldása
A több szóhoz rendelt kódok meghatározása
apa : 272; arc : 272; baj : 225; bal : 225; dal : 325; fal : 325; eb : 32;
fa : 32; eltesz : 358379; elvesz : 358379; fojt : 3658; folt : 3658; garas : 42727; harap : 42727;
hasas : 42727; guba : 4822; huba : 4822; haj : 425; hal : 425; kakas : 52527; kalap : 52527;
kap : 527; kar : 527; lap : 527; koma : 5662; lomb : 5662; kos : 567; lop : 567;
mar : 627; nap : 627; puha : 7842; ruha : 7842; raj : 725; rak : 725; szakad : 792523;
szalad : 792523; takar : 82527; vakar : 82527; tejes : 83537; teker : 83537; tesz : 8379; vesz : 8379;
tettes : 838837; tetves : 838837; vaj : 825; vak : 825;

A 9. feladat  megoldása
A leggazdagabb kód: 42727
3 szó tartozik hozzá: garas harap hasas
```

A kodok.txt szöveges fájl

```
22525
2276627
23
244
244632566
2525
2527
252728
252798
25227669
2562
257945
25828
264925
2692
272
...
```

2007 október: Foci

Perec város sportéletében fontos szerepet játszanak a fiatalok nagypályás labdarúgó mérkőzései. Tavasszal minden csapat minden csapattal pontosan egy mérkőzést játszott. A folyamatosan vezetett eredménylista azonban eltűnt, így csak a mérkőzések jegyzőkönyvei álltak rendelkezésre. A jegyzőkönyveket ismételten feldolgozták, ehhez első lépésként a *meccs.txt* állományba bejegyeztek néhány adatot. Önnek ezzel az állománnyal kell dolgoznia.

A *meccs.txt* állomány első sorában az állományban tárolt mérkőzések száma található. Alatta minden sorban egy-egy mérkőzés adatai olvashatók. Egy mérkőzést 7 adat ír le. Az első megadja, hogy a mérkőzést melyik fordulóban játszották le. A második a hazai, a harmadik a vendégcsapat góljainak száma a mérkőzés végén, a negyedik és ötödik a félidőben elért gólokat jelöli. A hatodik szöveg a hazai csapat neve, a hetedik a vendégcsapat neve. Az egyes adatokat egyetlen szóköz választja el egymástól. A sor végén nincs szóköz. A csapatok és a fordulók száma nem haladja meg a 20, a mérkőzések száma pedig a 400 értéket. Egy csapat sem rúgott meccsenként 9 gólnál többet. A csapatok neve legfeljebb 20 karakter hosszú, a névben nincs szóköz.

Például:

```
112
14 1 2 0 2 Agarak Ovatosak
5 4 0 1 0 Erosek Agarak
4 0 2 0 2 Ijedtek Hevesek
8 1 1 0 0 Ijedtek Nyulak
8 3 2 3 1 Lelkesek Bogarak
13 0 1 0 1 Fineszesek Csikosak
2 1 0 0 0 Pechesek Csikosak
1 4 0 2 0 Csikosak Kedvesek
9 2 0 0 0 Nyulak Lelkesek
6 0 2 0 0 Ovatosak Nyulak
```

Az 2. sor mutatja, hogy a 14. fordulóban az otthon játszó Agarakat az Óvatosak 2-1-re megverték úgy, hogy a félidőben már vezettek 2-0-ra.

Készítsen programot, amely az alábbi kérdésekre válaszol!
A program forráskódját mentse *foci* néven!
(A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: 3. feladat:). Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár!

1. Olvassa be a *meccs.txt* állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, az első 10 mérkőzés adatait jegyezze be a programba és dolgozzon azzal!
2. Kérje be a felhasználótól egy forduló számát, majd írja a képernyőre a bekért forduló mérkőzéseinek adatait a következő formában: `Edes-Savanyu: 2-0 (1-0)`! Soronként egy mérkőzést tüntessen fel! A különböző sorokban a csapatnevek ugyanazon a pozíción kezdődjenek!

   Például:

   ```
   Edes-Savanyu: 2-0 (1-0)
   Ijedtek-Hevesek: 0-2 (0-2)
   ...
   ```
3. Határozza meg, hogy a bajnokság során mely csapatoknak sikerült megfordítaniuk az állást a második félidőben! Ez azt jelenti, hogy a csapat az első félidőben vesztésre állt ugyan, de sikerült a mérkőzést megnyernie. A képernyőn soronként tüntesse fel a forduló sorszámát és a győztes csapat nevét!
4. Kérje be a felhasználótól egy csapat nevét, és tárolja el! A következő két feladat megoldásához ezt a csapatnevet használja! Ha nem tudta beolvasni, használja a Lelkesek csapatnevet!
5. Határozza meg, majd írja ki, hogy az adott csapat összesen hány gólt lőtt és hány gólt kapott! Például: `lőtt: 23 kapott: 12`
6. Határozza meg, hogy az adott csapat otthon melyik fordulóban kapott ki először és melyik csapattól! Ha egyszer sem kapott ki (ilyen csapat például a Bogarak), akkor „`A csapat otthon veretlen maradt.`” szöveget írja a képernyőre!
7. Készítsen statisztikát, amely megadja, hogy az egyes végeredmények hány alkalommal fordultak elő! Tekintse egyezőnek a fordított eredményeket (például 4-2 és 2-4)! A nagyobb számot mindig előre írja! Az elkészült listát a *stat.txt* állományban helyezze el!

   Például:

   ```
   2-1: 18 darab
   4-0: 2 darab
   2-0: 19 darab ...
   ```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Foci
 *
 * @author Klemand
 */

public class EmeltInfo2007okt {

	public static void main(String[] args) throws IOException {

		// 1. feladat
		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása a meccs.txt fájlból");

		BufferedReader beolvas = new BufferedReader(new FileReader("meccs.txt"));

		String sor = beolvas.readLine();

		int n = Integer.parseInt(sor.trim());
		// Ha esetleg zavaró szóköz lenne a végén, levágjuk

		int[] fordulo = new int[n]; // a forduló száma
		int[] hGol = new int[n]; // a hazi gólok száma
		int[] vGol = new int[n]; // a vendéggólok száma
		int[] hFel = new int[n]; // hazai gólok félidőben
		int[] vFel = new int[n]; // vendéggólok félidőben
		String[] hCsapat = new String[n]; // a hazai csapat neve
		String[] vCsapat = new String[n]; // a vendégcsapat neve

		String[] daraboltSor;
		int i;
		for (i = 1; i <= n; i++) {
			daraboltSor = beolvas.readLine().split(" ");
			fordulo[i - 1] = Integer.parseInt(daraboltSor[0]);
			hGol[i - 1] = Integer.parseInt(daraboltSor[1]);
			vGol[i - 1] = Integer.parseInt(daraboltSor[2]);
			hFel[i - 1] = Integer.parseInt(daraboltSor[3]);
			vFel[i - 1] = Integer.parseInt(daraboltSor[4]);
			hCsapat[i - 1] = daraboltSor[5];
			vCsapat[i - 1] = daraboltSor[6];
		}
		beolvas.close();

		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Egy billentyűzetről bekért forduló adatai");

		System.out.print("Kérem egy forduló sorszámát: ");

		Scanner sc1 = new Scanner(System.in);
		int ford = sc1.nextInt();

		for (i = 1; i <= n; i++) {
			if (fordulo[i - 1] == ford) {
				System.out.print(hCsapat[i - 1] + "-" + vCsapat[i - 1] + ": ");
				System.out.print(hGol[i - 1] + "-" + vGol[i - 1] + " (");
				System.out.println(hFel[i - 1] + "-" + vFel[i - 1] + ")");
			}
		}
		System.out.println("");
```

```
System.out.println("A 3. feladat megoldása");
System.out.println("A második félidőben fordító csapatok:");

for (i = 1; i <= n; i++) {
        if ((hGol[i - 1] - vGol[i - 1]) * (hFel[i - 1] - vFel[i - 1]) < 0) {
                System.out.print(fordulo[i - 1] + ". forduló: ");
                if (hGol[i - 1] > vGol[i - 1]) {
                        System.out.println(hCsapat[i - 1]);
                } else {
                        System.out.println(vCsapat[i - 1]);
                }
        }
}
System.out.println("");

System.out.println("A 4. feladat megoldása");
System.out.println("Egy csapat nevének bekérése: ");

System.out.print("Kérem egy csapat nevét: ");

Scanner sc2 = new Scanner(System.in);
String csapat = sc2.nextLine();

sc1.close();
sc2.close();

System.out.println("");

System.out.println("Az 5. feladat megoldása");
System.out.println("A beolvasott csapat lőtt és kapott góljainak meghatározása: ");
int lott = 0;
int kapott = 0;

for (i = 1; i <= n; i++) {
        if (hCsapat[i - 1].equals(csapat)) {
                lott += hGol[i - 1];
                kapott += vGol[i - 1];
        }
        if (vCsapat[i - 1].equals(csapat)) {
                lott += vGol[i - 1];
                kapott += hGol[i - 1];
        }
}
System.out.print(csapat + " ");
System.out.println("lőtt: " + lott + " kapott: " + kapott + "\n");

System.out.println("A 6. feladat megoldása");
System.out.println("A beolvasott csapat mikor és kitől kapott ki először otthon? ");

// Nem elég a tömlistában az első hazai vereség megkeresése, mert nincs
// fordulók szerint rendezve!
int mikor = 1000; // irreálisan nagy fordulószám megadása kezdőértéknek
int sorszam = 0;
for (i = 1; i <= n; i++) {
        if (hCsapat[i - 1].equals(csapat) && (hGol[i - 1] < vGol[i - 1])) {
                if (fordulo[i - 1] < mikor) {
                        mikor = fordulo[i - 1];
                        sorszam = i;
                }
        }
}
```

```
            if (mikor < 1000) {
                System.out.println("A(z) " + mikor + ". fordulóban a(z) ");
                System.out.println(vCsapat[sorszam - 1] + " csapattól.");
            } else {
                System.out.println("A csapat otthon veretlen maradt.");
            }
            System.out.println("");

            System.out.println("A 7. feladat megoldása");
            System.out.println("A végeredmények statisztikája:");

            String[] erLista = new String[n];
            int[] gyakorisag = new int[n];
            String aktEr;
            int erDb = 0;
            int j;
            for (i = 1; i <= n; i++) {
                j = 1;
                /*
                 * Az aktuális kód keresése az eredménylistában. Ha szerepel benne, növeljük a
                 * darabszámát, ha nem, akkor hozzávesszük a kódlistához.
                 */
                aktEr = erMeghat(hGol[i - 1], vGol[i - 1]);
                while ((j <= erDb) && !(aktEr.equals(erLista[j - 1]))) {
                    j++;
                }

                if (j <= erDb) {
                    gyakorisag[j - 1]++;
                } else {
                    erDb++;
                    erLista[erDb - 1] = aktEr;
                    gyakorisag[erDb - 1] = 1;
                }
            }

            // A statisztika fájlba írása
            PrintWriter kiir = new PrintWriter(new FileWriter("stat.txt"));

            for (i = 1; i <= erDb; i++) {
                kiir.println(erLista[i - 1] + ": " + gyakorisag[i - 1] + " darab");
            }

            kiir.close();
            System.out.println("A stat.txt fájl kiírása befejeződött. \n");
        }


        public static String erMeghat(int a, int b) {
            String er = "";
            er += Math.max(a, b) + "-" + Math.min(a, b);
            return er;
        }

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása a meccs.txt fájlból
A beolvasás megtörtént.

A 2. feladat megoldása
Egy billentyűzetről bekért forduló adatai
Kérem egy forduló sorszámát: 7
Kedvesek-Ovatosak: 1-3 (0-0)
Csikosak-Darabosak: 2-0 (0-0)
Bogarak-Ijedtek: 2-1 (0-1)
Hevesek-Agarak: 1-0 (0-0)
Jelmezesek-Pechesek: 1-2 (1-1)
Fineszesek-Lelkesek: 1-2 (1-0)
Nyulak-Mereszek: 1-0 (0-0)
Gyoztesek-Erosek: 2-1 (1-1)

A 3. feladat megoldása
A második félidőben fordító csapatok:
8. forduló: Jelmezesek
7. forduló: Bogarak
11. forduló: Ijedtek
14. forduló: Kedvesek
7. forduló: Lelkesek

A 4. feladat megoldása
Egy csapat nevének bekérése:
Kérem egy csapat nevét: Bogarak

Az 5. feladat megoldása
A beolvasott csapat lőtt és kapott góljainak meghatározása:
Bogarak lőtt: 18 kapott: 14

A 6. feladat megoldása
A beolvasott csapat mikor és kitől kapott ki először otthon?
A csapat otthon veretlen maradt.

A 7. feladat megoldása
A végeredmények statisztikája:
A stat.txt fájl kiírása befejeződött.
```

A stat.txt szövegfájl

```
2-1: 18 darab
4-0: 2 darab
2-0: 19 darab
1-1: 5 darab
3-2: 5 darab
1-0: 26 darab
3-1: 10 darab
4-2: 2 darab
3-0: 3 darab
4-3: 2 darab
0-0: 7 darab
3-3: 1 darab
2-2: 8 darab
4-1: 3 darab
5-1: 1 darab
```

2008 május: SMS

Esemes Ernő szenvedélyes SMS-küldő, ezért a MaMobil nevű cég tesztelésre kérte fel. Ehhez egy új, kézreálló telefont adnak, amelynek tesztüzemben egyetlen hátránya, hogy legfeljebb az először érkező 10 darab, egyenként legfeljebb 100 karakteres üzenetet tud eltárolni. Ha ettől több üzenet van, akkor azokat korlátlan számban a szolgáltató őrzi meg a hangpostához hasonlóan, tehát azokhoz csak bizonyos díj fejében juthat hozzá. Az üzenetek nem tartalmazhatnak ékezetes karaktereket.

Az *sms.txt* állomány első sorában az a k szám olvasható, amely megadja, hogy hány üzenet érkezett a készülékre a mai napon. Az érkező üzenetek száma legalább egy, de nem haladja meg a 100 darabot. Minden üzenethez 2 sor tartozik. Az első sor szerkezete a következő: először az érkezés órája (szám), érkezés perce (szám), telefonszám (pontosan 9 jegyű szám), a másodikban pedig az üzenet (legfeljebb 100 karakternyi szöveg) található. Az állományban az üzenetek számát követően k×2 sor szerepel. Az üzenetek érkezési idő szerint növekvően rendezettek.

Például:

```
30
9 11 123456789
Szia, mikor jossz?
9 13 434324223
Nem kerek ebedet!
9 14 434324223
Hova menjek erted?
9 20 123456789
Hozd el a mintas pulcsimat!
9 21 434324223
Nyertünk a palyazaton! …
```

Készítsen programot *sms* néven, amely az alábbi kérdésekre válaszol! Ügyeljen arra, hogy a program forráskódját a megadott helyre mentse!

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát! (Például `3. feladat:`)

1. Olvassa be az *sms.txt* állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, akkor a benne található adatok közül az első tíz üzenet adatait jegyezze be a programba, s úgy oldja meg a feladatokat!
2. A fájlban tárolt utolsó üzenet érkezésekor melyik üzenet a legfrissebb a telefon memóriájában? Írja az üzenet szövegét a képernyőre!
3. Adja meg a leghosszabb és a legrövidebb üzenetek adatait! Ha több azonos hosszúságú üzenet van, akkor elegendő csak egyet-egyet megadnia! A képernyőn óra, perc, telefonszám, üzenet formában jelenítse meg az adatokat!
4. Készítsen karakterhossz szerinti statisztikát: `1-20`, `21-40`, `41-60`, `61-80`, `81-100`! Az intervallumok mellé a hozzájuk tartozó üzenetek darabszámát írja, mint eredményt a képernyőre!

5. Ha Ernő minden óra 0. percében elolvasná a memóriában lévő üzeneteket (az éppen ekkor érkező üzeneteket nem látja), majd ki is törölné, akkor hány olyan üzenet lenne, amelynek elolvasásához fel kellene hívnia a szolgáltatót? Írja ezt a számot a képernyőre! (Az üzeneteket először 1, utoljára 24 órakor olvassa el.)
6. Ernő barátnője gyakran küld sms-t az 123456789-es számról. Mennyi volt a leghosszabb idő, amennyi eltelt két üzenete között? Ha legfeljebb 1 üzenet érkezett tőle, akkor írja ki, hogy „nincs elegendő üzenet”, egyébként pedig adja meg a leghosszabb időtartamot óra perc alakban!
7. Egy üzenet véletlenül késett. Olvassa be a billentyűzetről ennek az sms-nek az adatait, majd tárolja el a memóriában a többihez hasonlóan!
8. Az *smski.txt* állományban készítsen egy listát az üzenetekről telefonszám szerinti csoportosításban, telefonszám szerint növekvő sorrendben! Egy csoporthoz tartozó első sorban a feladó telefonszáma szerepeljen! Az alatta lévő sorokban a feladás ideje, majd a tőle újabb szóközzel elválasztva az üzenet szövege szerepeljen!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * SMS
 *
 * @author Klemand
 */

public class EmeltInfo2008maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása az sms.txt fájlból");

		BufferedReader beolvas = new BufferedReader(new FileReader("sms.txt"));

		String sor;
		sor = beolvas.readLine();

		int n = Integer.parseInt(sor.trim());
		// Ha esetleg felesleges szóköz lenne a végén, levágjuk

		int[] ora = new int[n + 1]; // Lesz egy elkallódott üzenet!
		int[] perc = new int[n + 1];
		String[] telSzam = new String[n + 1];
		String[] uzenet = new String[n + 1];

		int i, j;
		String[] daraboltSor;

		for (i = 1; i <= n; i++) {
			daraboltSor = beolvas.readLine().split(" ");
			ora[i - 1] = Integer.parseInt(daraboltSor[0]);
			perc[i - 1] = Integer.parseInt(daraboltSor[1]);
			telSzam[i - 1] = daraboltSor[2];
			uzenet[i - 1] = beolvas.readLine();
		}
		beolvas.close();

		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Az utolsó üzenet érkezésekor a legfrissebb üzenet a memóriában:");
		int C = 10; // kapacitás

		if (n < C) {
			System.out.println(uzenet[n - 1]);
		} else {
			System.out.println(uzenet[C - 1]);
		}

		System.out.println("");

		System.out.println("A 3. feladat megoldása");
		System.out.println("A leghosszabb és a legrövidebb üzenet: ");

		int lh = 1;
		int lr = 1;
```

```
for (i = 1; i <= n; i++) {
        if (uzenet[i - 1].length() > uzenet[lh - 1].length()) {
                lh = i;
        }
        if (uzenet[i - 1].length() < uzenet[lr - 1].length()) {
                lr = i;
        }
}

System.out.println("A leghosszabb üzenet adatai: ");
System.out.print(ora[lh - 1] + ", " + perc[lh - 1] + ", ");
System.out.println(telSzam[lh - 1] + ", " + uzenet[lh - 1]);
System.out.println("A legrövidebb üzenet adatai: ");
System.out.print(ora[lr - 1] + ", " + perc[lr - 1] + ", ");
System.out.println(telSzam[lr - 1] + ", " + uzenet[lr - 1]);
System.out.println("");

System.out.println("A 4. feladat megoldása");
System.out.println("Karakterhossz szerinti statisztika");

int db01 = 0;
int db21 = 0;
int db41 = 0;
int db61 = 0;
int db81 = 0;

for (i = 1; i <= n; i++) {

        if (uzenet[i - 1].length() > 80) {
                db81++;
        } else if (uzenet[i - 1].length() > 60) {
                db61++;
        } else if (uzenet[i - 1].length() > 40) {
                db41++;
        } else if (uzenet[i - 1].length() > 20) {
                db21++;
        } else {
                db01++;
        }

}
System.out.println(" 1 - 20: " + db01);
System.out.println("21 - 40: " + db21);
System.out.println("41 - 60: " + db41);
System.out.println("61 - 80: " + db61);
System.out.println("81 -100: " + db81);
System.out.println("");

System.out.println("Az 5. feladat megoldása");
System.out.println("Törlés minden órában:");

System.out.println("Az egyes órákban érkező sms-eket");
System.out.println("a feladási órákkal indexelt vektorban számoljuk meg.");

int[] orakDb = new int[24];

for (i = 1; i <= 24; i++) { // Kezdőérték beállítás
        orakDb[i - 1] = 0;
}

for (i = 1; i <= n; i++) {
        orakDb[ora[i - 1]]++;
}
```

```
int szolgDb = 0;

for (i = 1; i <= 24; i++) {
        if (orakDb[i - 1] > 10) {
                szolgDb += orakDb[i - 1] - 10;
        }
}

System.out.println("A szolgáltatónál maradó üzenetek száma: " + szolgDb + "\n");

System.out.println("A 6. feladat megoldása");
System.out.print("Ernő és a barátnője");

int[] bno = new int[100];
int bnoDb = 0;

for (i = 1; i <= n; i++) {
        if (telSzam[i - 1].equals("123456789")) {
                bnoDb++;
                bno[bnoDb - 1] = ora[i - 1] * 60 + perc[i - 1];
        }
}

int leghosszabb = -1;

if (bnoDb < 2) {
        System.out.println("Nincs elegendő üzenet.");
} else {
        leghosszabb = bno[1] - bno[0];
        for (i = 3; i <= bnoDb; i++) {
                if (bno[i - 1] - bno[i - 2] > leghosszabb) {
                        leghosszabb = bno[i - 1] - bno[i - 2];
                }
        }
        System.out.print("A leghosszabb idő, amíg Ernőnek a barátnője nem küldött sms-t: ");
        System.out.println(leghosszabb / 60 + " óra " + leghosszabb % 60 + " perc ");
}

System.out.println("");

System.out.println("A 7. feladat megoldása");
System.out.print("Egy elkallódott SMS beolvasása a billentyűzetről ");
System.out.println("és hozzáfűzése a tömbhöz");

System.out.println("Kérem az SMS adatait!");

Scanner sc = new Scanner(System.in);

System.out.println("Kérem az órát, a percet és a telefonszámot szóközökkel elválasztva!");

daraboltSor = sc.nextLine().split(" ");

/*
 * Ha a véletlenül begépelt többszörös szóközöket is kezelni szeretnénk:
 * daraboltSor = sc.nextLine().trim().split("\\s+");
 */

ora[n] = Integer.parseInt(daraboltSor[0]);
perc[n] = Integer.parseInt(daraboltSor[1]);
telSzam[n] = daraboltSor[2];

System.out.println("Kérem az üzenetet! ");
uzenet[n] = sc.nextLine();

sc.close();
```

```
                System.out.println("\nA 8. feladat megoldása");
                System.out.print("Lista az üzenetekről telefonszám szerinti csoportosításban ");
                System.out.println("az smski.txt állományban");

                /*
                 * A lista rendezése telefonszám szerint. Buborékos rendezést alkalmazunk,
                 * ami megőrzi az időpont szerinti rendezést.
                 */
                String StringAsztal;
                int intAsztal;

                for (i = n; i >= 1; i--) { // Már n + 1 üzenet van a tömbben!
                        for (j = 1; j <= i; j++)
                                if (telSzam[j - 1].compareTo(telSzam[j]) > 0) {

                                        intAsztal = ora[j - 1];
                                        ora[j - 1] = ora[j];
                                        ora[j] = intAsztal;

                                        intAsztal = perc[j - 1];
                                        perc[j - 1] = perc[j];
                                        perc[j] = intAsztal;

                                        StringAsztal = telSzam[j - 1];
                                        telSzam[j - 1] = telSzam[j];
                                        telSzam[j] = StringAsztal;

                                        StringAsztal = uzenet[j - 1];
                                        uzenet[j - 1] = uzenet[j];
                                        uzenet[j] = StringAsztal;
                                }
                }

                PrintWriter kivitel;
                kivitel = new PrintWriter(new FileWriter("smski.txt"));

                String telszam = "-1";
                // irreális kezdőérték a telefonszámok élre írásához

                for (i = 1; i <= n + 1; i++) {
                        if (!telSzam[i - 1].equals(telszam)) {
                                kivitel.println(telSzam[i - 1]);
                                telszam = telSzam[i - 1];
                        }
                        kivitel.println(ora[i - 1] + " " + perc[i - 1] + " " + uzenet[i - 1]);
                }

                kivitel.close();
                System.out.println("Az smski.txt fájl kiírása befejeződött. \n");
        }

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása az sms.txt fájlból
A beolvasás megtörtént.

A 2. feladat megoldása
Az utolsó üzenet érkezésekor a legfrissebb üzenet a memóriában:
Erdekli Ont egy telefon? A vezetek nelkuli telefon most a legolcsobb!

A 3. feladat megoldása
A leghosszabb és a legrövidebb üzenet:
A leghosszabb üzenet adatai:
9, 40, 434325432, A nyelvvizsgadra mennyi potlekot kapsz? Nekem meg nem fizettek egy fillert sem. :-(
A legrövidebb üzenet adatai:
11, 36, 434324223, Kesz a kocsi!

A 4. feladat megoldása
Karakterhossz szerinti statisztika
 1 - 20: 5
21 - 40: 17
41 - 60: 2
61 - 80: 5
81 -100: 1

Az 5. feladat megoldása
Törlés minden órában:
Az egyes órákban érkező sms-eket
a feladási órákkal indexelt vektorban számoljuk meg.
A szolgáltatónál maradó üzenetek száma: 3

A 6. feladat megoldása
Ernő és a barátnőjeA leghosszabb idő, amíg Ernőnek a barátnője nem küldött sms-t: 1 óra 27 perc

A 7. feladat megoldása
Egy elkallódott SMS beolvasása a billentyűzetről és hozzáfűzése a tömbhöz
Kérem az SMS adatait!
Kérem az órát, a percet és a telefonszámot szóközökkel elválasztva!
12 38 123456780
Kérem az üzenetet!
Képzeld, a különc műlábú gépírónő ma férjhez megy a tündérszárnyú kígyóbűvölőhöz!

A 8. feladat megoldása
Lista az üzenetekről telefonszám szerinti csoportosításban az smski.txt állományban
Az smski.txt fájl kiírása befejeződött.
```

Az smski.txt szoveges fájl

```
123456780
16 42 Képzeld, a különc műlábú gépírónő ma férjhez megy a tündérszárnyú kígyóbűvölőhöz!
123456789
9 11 Szia, mikor jossz?
9 20 Hozd el a mintas pulcsimat!
10 8 Hol vagy mar olyan sokaig? Varlak!
11 14 Este szinhazba megyunk, ugye tudod?
11 21 Holnap jonnek Agiek!
12 48 A szinhaz ugrott, csotores van!
231287556
10 25 Tibi megjott, majd hivd fel delben!
343567452
9 16 Nem erek oda idoben. Hivd fel a fonokot es ments ki!
434324223
9 13 Nem kerek ebedet!
9 14 Hova menjek erted?
9 21 Nyertünk a palyazaton!
9 45 A gep nem bootol be. Aramot kap, de a monitoron nem jelenik meg semmi.
9 53 Elvesztetted a fogadast! :-)
11 22 Utalhatod a penzt a lakasert!
11 36 Kesz a kocsi!
11 50 No mi van, akarsz focizni?
12 3 Lesz egy londoni ut, erdekel?
434325432
9 40 A nyelvvizsgadra mennyi potlekot kapsz? Nekem meg nem fizettek egy fillert sem. :-(
434325632
9 46 Hova tetted a palyazati urlapot? Mar fel oraja keressuk.
435435345
11 19 A megrendelt konyve megerkezett.
454343545
10 12 Add fel postan meg ma a palyazatot!
545345345
11 3 Erdekli Ont egy telefon? A vezetek nelkuli telefon most a legolcsobb!
545432542
9 51 Erdekli Ont egy telefon? A vezetek nelkuli telefon most a legolcsobb!
545453345
11 1 Potyara jottem, az ugyfel nem volt itt.
565643244
10 44 Erdekli Ont egy telefon? A vezetek nelkuli telefon most a legolcsobb!
654647445
11 29 Erdekli Ont egy telefon? A vezetek nelkuli telefon most a legolcsobb!
853556565
11 4 Sot, az urge visszalepett.
854655455
11 11 Hova szallitsam a butort?
11 13 Mikor kezdodik a meccs? A jegyet veszed?
12 44 Hany napra kersz szallast?
```

2008 október: Robot

Gáborék iskolai szakkörön robotot építenek. Már elkészítettek egy olyan változatot, amelyik sík terepen kellő pontossággal vezérelhető. A robot a memóriájába előre betáplált programok egyikét hajtja végre. A robot jelenleg csak az E, K, D, N utasításokat érti, amelyek a négy égtáj (sorrendben: észak, kelet, dél, nyugat) irányában tett 1 centiméteres elmozdulást eredményezik.

A robotba táplált programokat a *program.txt* állományban rögzítettük. Az állomány első sorában a betáplált programok száma található, amely legfeljebb 100. Alatta soronként egy-egy program olvasható. Egy sor legfeljebb 200 karakter hosszúságú, benne az E, K, D, N karakterek mint utasítások találhatók. A sorok nem tartalmaznak szóközt.

Például:
program.txt

```
12
ENNNDKENDND
ENNDDDDENDENDEEDDNNKED ...
```

A 2. sorban az első betáplált program utasításai vannak.

Készítsen programot, amely az alábbi kérdésekre válaszol! A program forráskódját *robot* néven mentse!

Minden részfeladat megoldása előtt írja a képernyőre a feladat sorszámát! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár (például `2. feladat: Kérem az utasítássor sorszámát!`)!

1. Olvassa be a *program.txt* állományban talált adatokat, s azok felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, az állomány első 10 sorának adatait jegyezze be a programba és dolgozzon azzal!
2. Kérje be egy utasítássor számát, majd írja a képernyőre, hogy:
   a. Egyszerűsíthető-e az utasítássorozat! Az `egyszerűsíthető`, illetve `nem egyszerűsíthető` választ írja a képernyőre! (Egy utasítássort egyszerűsíthetőnek nevezünk, ha van benne két szomszédos, ellentétes irányt kifejező utasításpár, hiszen ezek a párok elhagyhatók. Ilyen ellentétes utasításpár az ED, DE, KN, NK.)
   b. Az utasítássor végrehajtását követően legkevesebb mennyi E vagy D és K vagy N utasítással lehetne a robotot a kiindulási pontba visszajuttatni! A választ a következő formában jelenítse meg:
   `3 lépést kell tenni az ED, 4 lépést a KN tengely mentén.`
   c. Annak végrehajtása során hányadik lépést követően került (légvonalban) legtávolabb a robot a kiindulási ponttól és mekkora volt ez a távolság! A távolságot a lépés sorszámát követően 3 tizedes pontossággal írja a képernyőre!

3. A robot a mozgáshoz szükséges energiát egy beépített akkuból nyeri. A robot 1 centiméternyi távolság megtételéhez 1 egység, az irányváltásokhoz és az induláshoz 2 egység energiát használ. Ennek alapján az EKK utasítássor végrehajtásához 7 egység energia szükséges. A szakkörön használt teljesen feltöltött kis kapacitású akkuból 100, a nagykapacitásúból 1000 egységnyi energia nyerhető ki. Adja meg azon utasítássorokat, amelyek végrehajtásához a teljesen feltöltött kis kapacitású akku is elegendő! Írja a képernyőre egymástól szóközzel elválasztva az utasítássor sorszámát és a szükséges energia mennyiségét! Minden érintett utasítássor külön sorba kerüljön!
4. Gáborék továbbfejlesztették az utasításokat értelmező programot. Az új, jelenleg még tesztelés alatt álló változatban a több, változatlan irányban tett elmozdulást helyettesítjük az adott irányban tett elmozdulások számával és az irány betűjével. Tehát például a DDDKDD utasítássor leírható rövidített 3DK2D formában is. Az önállóan álló utasításnál az 1-es számot nem szabad kiírni! Hozza létre az *ujprog.txt* állományt, amely a *program.txt* állományban foglalt utasítássorozatokat az új formára alakítja úgy, hogy az egymást követő azonos utasításokat minden esetben a rövidített alakra cseréli! Az *ujprog.txt* állományba soronként egy utasítássor kerüljön, a sorok ne tartalmazzanak szóközt!
5. Sajnos a tesztek rámutattak arra, hogy a program új verziója még nem tökéletes, ezért vissza kell térni az utasítássorok leírásának régebbi változatához. Mivel a szakkörösök nagyon bíztak az új változatban, ezért néhány utasítássort már csak ennek megfelelően készítettek el. Segítsen ezeket visszaírni az eredeti formára! Az ismétlődések száma legfeljebb 200 lehet! Kérjen be egy új formátumú utasítássort, majd írja a képernyőre régi formában!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Robot
 *
 * @author Klemand
 */

public class EmeltInfo2008okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása a program.txt fájlból ");

		BufferedReader beolvas = new BufferedReader(new FileReader("program.txt"));

		String sor;
		sor = beolvas.readLine();

		int n = Integer.parseInt(sor.trim());
		// Ha esetleg felesleges szóköz lenne a végén, levágjuk

		String[] program = new String[n];

		int i;
		for (i = 1; i <= n; i++) {
			program[i - 1] = beolvas.readLine();
		}
		beolvas.close();

		System.out.println("A beolvasás megtörtént.\n ");

		System.out.println("A 2. feladat megoldása");

		System.out.print("Kérem egy program sorszámát (max. " + n + "): ");

		Scanner sc1 = new Scanner(System.in);

		int sorszam = sc1.nextInt();
		System.out.println("");

		System.out.println("A 2. a.  feladat megoldása");

		String pr = program[sorszam - 1]; // az aktuális program

		if (pr.contains("ED") || pr.contains("DE") || pr.contains("KN") || pr.contains("NK")) {
			System.out.println("A program egyszerűsíthető.");
		} else {
			System.out.println("A program nem egyszerűsíthető.");
		}
		System.out.println("");
```

```
System.out.println("A 2. b. és 2. c. feladatok megoldása");

int hossz = pr.length();
int EDb = 0;
int DDb = 0;
int KDb = 0;
int NDb = 0;
String irany;
int max = 0;
double aktTav;
double maxTav = 0;

for (i = 1; i <= hossz; i++) {

        irany = pr.substring(i - 1, i);

        if (irany.equals("E")) {
                EDb++;
        } else if (irany.equals("D")) {
                DDb++;
        } else if (irany.equals("K")) {
                KDb++;
        } else {
                NDb++;
        }

        aktTav = Math.sqrt(Math.pow(EDb - DDb, 2) + Math.pow(KDb - NDb, 2));

        if (aktTav > maxTav) {
                max = i;
                maxTav = aktTav;
        }
}

System.out.print("Visszaút : ");
System.out.print(Math.abs(EDb - DDb) + " lépést kell tenni ED, ");
System.out.println("és " + Math.abs(KDb - NDb) + " lépést KN irányban.\n");
System.out.print(max + ". lépés után a legnagyobb a távolság a kiindulási ponttól: ");
System.out.printf("%.3f", maxTav);
System.out.println(" cm. \n");

System.out.println("A 3. feladat megoldása");
System.out.println("A következő programok végrehajtásához elegendő a kis akku: ");
int energiaSzuks;

for (i = 1; i <= n; i++) {
        energiaSzuks = energiaSzuksMeghat(program[i - 1]);
        if (energiaSzuks < 100) {
                System.out.println(i + " " + energiaSzuks);
        }
}

System.out.println("\nA 4. feladat megoldása");
System.out.println("Az új programok kiírása a ujprog.txt állományba");

PrintWriter kiir = new PrintWriter(new FileWriter("ujprog.txt"));

for (i = 1; i <= n; i++) {
        kiir.println(ujProgramMeghat(program[i - 1]));
}

kiir.close();
System.out.println("Az ujprog.txt fájl kiírása befejeződött. \n");
```

```
        System.out.println("Az 5. feladat megoldása");
        System.out.println("Billentyűzetről bekért új program visszaalakítása");

        System.out.println("Kérek egy új típusú programot)!");
        System.out.println("Csak számok (max. 200) és E, D, K, N betűk lehetnek szóköz nélkül:");

        Scanner sc2 = new Scanner(System.in);

        String ujpr;
        ujpr = sc2.nextLine().trim().toUpperCase();
        System.out.println("Régi alakja: " + visszaFejt(ujpr));

        sc1.close();
        sc2.close();
        System.out.println("");
}

public static int energiaSzuksMeghat(String program) {
        int energia = 3;
        // indulás és első lépés
        for (int j = 2; j <= program.length(); j++) {
                energia++;
                if (!program.substring(j - 1, j).equals(program.substring(j - 2, j - 1))) {
                        energia += 2;
                }
        }
        return energia;
}

public static String ujProgramMeghat(String program) {

        String ujpr = "";
        String irany = program.substring(0, 1);
        int db = 1;
        int j;
        for (j = 2; j <= program.length(); j++) {
                if (program.substring(j - 1, j).equals(irany)) {
                        db++;
                } else {
                        // A fordulás előtti irány kiírása
                        if (db > 1) {
                                ujpr += db + irany;
                        } else {
                                ujpr += irany;
                        }
                        irany = program.substring(j - 1, j);
                        db = 1;
                }
        }

        if (db > 1) { // Lezárás
                ujpr += (db + irany);
        } else {
                ujpr += irany;
        }
        return ujpr;
}
```

```
        public static String visszaFejt(String ujpr) {
                String regipr = "";
                int szamErtek = 0;

                for (int j = 1; j <= ujpr.length(); j++) {
                        if ("0123456789".contains(ujpr.substring(j - 1, j))) {
                                szamErtek = 10 * szamErtek + Integer.parseInt(ujpr.substring(j - 1, j));
                        } else {
                                if (szamErtek > 0) {
                                        for (int k = 1; k <= szamErtek; k++) {
                                                regipr += ujpr.substring(j - 1, j);
                                        }
                                        szamErtek = 0;
                                } else {
                                        regipr += ujpr.substring(j - 1, j);
                                }
                        }
                }
                return regipr;
        }

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása a program.txt fájlból
A beolvasás megtörtént.

A 2. feladat megoldása
Kérem egy program sorszámát (max. 13): 10

A 2. a.  feladat megoldása
A program egyszerűsíthető.

A 2. b. és 2. c. feladatok megoldása
Visszaút : 8 lépést kell tenni ED, és 11 lépést KN irányban.

159. lépés után a legnagyobb a távolság a kiindulási ponttól: 13,601 cm.

A 3. feladat megoldása
A következő programok végrehajtásához elegendő a kis akku:
1 29
2 52
3 84

A 4. feladat megoldása
Az új programok kiírása a ujprog.txt állományba
Az ujprog.txt fájl kiírása befejeződött.

Az 5. feladat megoldása
Billentyűzetről bekért új program visszaalakítása
Kérek egy új típusú programot)!
Csak számok (max. 200) és E, D, K, N betűk lehetnek szóköz nélkül:
E2N4DENDEND2E2D2NKED
Régi alakja: ENNDDDDENDENDEEDDNNKED
```

A program.txt forrásfájl

```
13
ENNNDKENDND
ENNDDDDENDENDEEDDNNKED
EKDNEKDNEKDNEKDNEKDNEKDNEKDN
...
```

Az ujprogram.txt szöveges fájl

```
E3NDKENDND
E2N4DENDEND2E2D2NKED
EKDNEKDNEKDNEKDNEKDNEKDNEKDN
...
```

2009 május: Lift

A Madárház Kft. toronyházak építésével foglalkozik. Jelenleg a Csúcs Rt. 100 szintes szerkezetkész épületén kezdték meg a belső szerelési műveleteket. Az egyes szerelőcsapatok naponta többször változtatják helyüket. Ha az új munkaterület egy másik emeleten van, akkor – a biztonsági előírások miatt – lifttel kell menniük. A házban egyetlen lift működik, amelynek igénybevételét az egyes csapatok a célszint megadásával jelezhetik. A lift az igényeket a jelzés sorrendjében szolgálja ki, és egyszerre csak egy csapatot szállít. A csapatok mozgását a 9 és 14 óra közötti intervallumban követjük nyomon. Ez az intervallum a munkaidőnek csak egy része, tehát a csapatok már dolgoznak valamelyik szinten, de 9 órakor teljesítetlen kérés nincs és a lift szabad.

A lifthasználati igényeket az *igeny.txt* állomány tartalmazza. Első sorában a szintek száma (legfeljebb 100), a második sorban a csapatok száma (legfeljebb 50), a harmadik sorban pedig az igények száma (legfeljebb 100) olvasható. A negyedik sortól kezdve soronként egy-egy igény szerepel a jelzés sorrendjében. Egy igény hat számból áll: az első három szám az időt adja meg (óra, perc, másodpercszám sorrendben), a negyedik a csapat sorszáma, az ötödik az induló-, a hatodik a célszint sorszáma. Az egyes számokat pontosan egy szóköz választja el egymástól.

Például:

igeny.txt

```
100
10
55
9 7 11 7 6 22
9 10 30 8 18 2
9 11 0 5 12 20
…
```

A 4. sor megmutatja, hogy 9 óra 7 perc 11 másodperckor a 7. csapat igényelt liftet, hogy a 6. szintről a 22. szintre eljusson.

Készítsen programot, amely az alábbi kérdésekre válaszol! A program forráskódját *lift* néven mentse! Ügyeljen arra, hogy programjának minden helyes tartalmú bemeneti állomány esetén működnie kell!

Minden részfeladat megoldása előtt írja a képernyőre a feladat sorszámát! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár (például a 2. feladat esetén: „`2. feladat Kérem a lift indulási helyét!`”)!
A képernyőn megjelenített üzenetek esetén az ékezetmentes kiírás is elfogadott.

1. Olvassa be az *igeny.txt* állományban talált adatokat, s azok felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, az első 8 igényhez tartozó adatokat jegyezze be a programba és dolgozzon azzal!
2. Tudjuk, hogy a megfigyelés kezdetén a lift éppen áll. Kérje be a felhasználótól, hogy melyik szinten áll a lift, és a további részfeladatok megoldásánál ezt vegye figyelembe! Ha a beolvasást nem tudja elvégezni, használja az *igény.txt* fájlban az első igény induló szintjét!
3. Határozza meg, hogy melyik szinten áll majd a lift az utolsó kérés teljesítését követően! Írja képernyőre a választ a következőhöz hasonló formában:
   „`A lift a 33. szinten áll az utolsó igény teljesítése után.`” !

4. Írja a képernyőre, hogy a megfigyelés kezdete és az utolsó igény teljesítése között melyik volt a legalacsonyabb és melyik a legmagasabb sorszámú szint, amelyet a lift érintett!

5. Határozza meg, hogy hányszor kellett a liftnek felfelé indulnia utassal és hányszor utas nélkül! Az eredményt jelenítse meg a képernyőn!

6. Határozza meg, hogy mely szerelőcsapatok nem vették igénybe a liftet a vizsgált intervallumban! A szerelőcsapatok sorszámát egymástól egy-egy szóközzel elválasztva írja a képernyőre!

7. Előfordul, hogy egyik vagy másik szerelőcsapat áthágja a szabályokat, és egyik szintről gyalog megy a másikra. (Ezt onnan tudhatjuk, hogy más emeleten igényli a liftet, mint ahova korábban érkezett.) Generáljon véletlenszerűen egy létező csapatsorszámot! (Ha nem jár sikerrel, dolgozzon a 3. csapattal!) Határozza meg, hogy a vizsgált időszak igényei alapján lehet-e egyértelműen bizonyítani, hogy ez a csapat vétett a szabályok ellen! Ha igen, akkor adja meg, hogy melyik két szint közötti utat tették meg gyalog, ellenkező esetben írja ki a Nem bizonyítható szabálytalanság szöveget!

8. A munkák elvégzésének adminisztrálásához minden csapatnak egy blokkoló kártyát kell használnia. A kártyára a liftben elhelyezett blokkolóóra rögzíti az emeletet, az időpontot. Ennek a készüléknek a segítségével kell megadni a munka kódszámát és az adott munkafolyamat sikerességét. A munka kódja 1 és 99 közötti egész szám lehet. A sikerességet a „befejezett” és a „befejezetlen” szavakkal lehet jelezni.
   Egy műszaki hiba folytán az előző feladatban vizsgált csapat kártyájára az általunk nyomon követett időszakban nem került bejegyzés. Ezért a csapatfőnöknek a műszak végén pótolnia kell a hiányzó adatokat. Az *igeny.txt* állomány adatait felhasználva írja a képernyőre időrendben, hogy a vizsgált időszakban milyen kérdéseket tett fel az óra, és kérje be az adott válaszokat a felhasználótól! A pótlólag feljegyzett adatokat írja a *blokkol.txt* állományba! A *blokkol.txt* állomány tartalmát az alábbi sorok mintájára alakítsa ki:

```
Befejezés ideje: 9:23:11
Sikeresség: befejezett
-----
Indulási emelet: 9
Célemelet: 11
Feladatkód: 23
Befejezés ideje: 10:43:22
Sikeresség: befejezetlen
-----
Indulási emelet: 11
Célemelet: 6
Feladatkód: 6
…
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Lift
 *
 * @author Klemand
 */

public class EmeltInfo2009maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az igeny.txt fájl beolvasása ");

		BufferedReader beolvas = new BufferedReader(new FileReader("igeny.txt"));

		int szintDb = Integer.parseInt(beolvas.readLine());
		int csapatDb = Integer.parseInt(beolvas.readLine());
		int igenyDb = Integer.parseInt(beolvas.readLine());

		int[] ora = new int[igenyDb];
		int[] perc = new int[igenyDb];
		int[] mp = new int[igenyDb];
		int[] csapat = new int[igenyDb];
		int[] indSzint = new int[igenyDb]; // indulószint
		int[] celSzint = new int[igenyDb]; // célszint

		int i;
		String[] daraboltSor;

		for (i = 1; i <= igenyDb; i++) {
			daraboltSor = beolvas.readLine().split(" ");
			ora[i - 1] = Integer.parseInt(daraboltSor[0]);
			perc[i - 1] = Integer.parseInt(daraboltSor[1]);
			mp[i - 1] = Integer.parseInt(daraboltSor[2]);
			csapat[i - 1] = Integer.parseInt(daraboltSor[3]);
			indSzint[i - 1] = Integer.parseInt(daraboltSor[4]);
			celSzint[i - 1] = Integer.parseInt(daraboltSor[5]);
		}
		beolvas.close();

		System.out.println("\nA 2. feladat megoldása");
		System.out.println("A lift kezdeti helyzetének beolvasása");

		Scanner sc1 = new Scanner(System.in);

		System.out.print("Kérem az indulási szintet, ahol a lift 9 órakor áll ");
		System.out.print("(max. " + szintDb + "): ");

		int szint0 = sc1.nextInt();

		System.out.println("\nA 3. feladat megoldása");
		System.out.println("A lift végső helyzetének meghatározása");

		System.out.print("A lift a(z) " + celSzint[igenyDb - 1]);
		System.out.println(". szinten áll az utolsó igény teljesítése után. \n");
```

```
System.out.println("A 4. feladat megoldása");
System.out.println("A lift szélső helyzeteinek meghatározása");

int aktMin;
int aktMax;
int legalacsonyabb = szint0;
int legmagasabb = szint0;

for (i = 1; i <= igenyDb; i++) {
        aktMin = Math.min(indSzint[i - 1], celSzint[i - 1]);
        aktMax = Math.max(indSzint[i - 1], celSzint[i - 1]);
        if (aktMin < legalacsonyabb) {
                legalacsonyabb = aktMin;
        }
        if (aktMax > legmagasabb) {
                legmagasabb = aktMax;
        }
}
System.out.println("A legalacsonyabb szint: " + legalacsonyabb);
System.out.println("A legmagasabb szint: " + legmagasabb);
System.out.println("");

System.out.println("Az 5. feladat megoldása");
System.out.println("Felfelé indulások száma utassal és utas nélkül");

int szint = szint0;
int utassal = 0;
int uresen = 0;

for (i = 1; i <= igenyDb; i++) {
        if (indSzint[i - 1] > szint) {
                uresen++;
        }
        if (celSzint[i - 1] > indSzint[i - 1]) {
                utassal++;
        }
        szint = celSzint[i - 1];
}
System.out.println("A liftnek " + utassal + " esetben kellett felfelé menni utassal, ");
System.out.println("és " + uresen + " esetben utas nélkül. \n ");

System.out.println("A 6. feladat megoldása");
System.out.println("A következő szerelőcsapatok nem vették igénybe a liftet: ");

boolean[] csapatIgenyles = new boolean[csapatDb];

for (i = 1; i <= csapatDb; i++) {
        csapatIgenyles[i - 1] = false; // Kezdőérték beállítás
}

// Igénylés esetén igazra állítjuk az igénylő csapat igénylését
int aktCsapat;

for (i = 1; i <= igenyDb; i++) {
        aktCsapat = csapat[i - 1];
        csapatIgenyles[aktCsapat - 1] = true;
}

// Ahol a csapatIgénylés hamis maradt
for (i = 1; i <= csapatDb; i++) {
        if (!csapatIgenyles[i - 1]) {
                System.out.print(i + " ");
        }
}
System.out.println("\n");
```

```
System.out.println("A 7. feladat megoldása");
System.out.println("Szabálytalankodott-e egy véletlenszerűen kiválasztott csapat?");

System.out.print("A véletlenszerűen kiválasztott csapat sorszáma: ");

int sorszam;
sorszam = (int) (csapatDb * Math.random()) + 1;

System.out.println(sorszam);

// Kigyűjtjük a vizsgált csapat mozgását
int[] ind = new int[igenyDb];
int[] erk = new int[igenyDb];

int mozgasDb = 0;

for (i = 1; i <= igenyDb; i++) {
        if (csapat[i - 1] == sorszam) {
                mozgasDb++;
                ind[mozgasDb - 1] = indSzint[i - 1]; // indulás
                erk[mozgasDb - 1] = celSzint[i - 1]; // érkezés
        }
}

boolean szabalytalankodott = false;
// tiszta lappal indul a csapat
int szInd = -1;
int szErk = -1;
// irreális kezdőértékek a szabálytalan mozgásra

i = 2;
while (i <= mozgasDb && !szabalytalankodott) {
        if (ind[i - 1] != erk[i - 2]) {
                szabalytalankodott = true;
                szInd = erk[i - 2];
                szErk = ind[i - 1];
        }
        i++;
}

if (szabalytalankodott) {
        System.out.println("Bizonyíthatóan szabálytalankodtak.");
        System.out.println("Gyalog közlekedtek a(z) " + szInd + " és " + szErk + " szintek között.");
} else {
        System.out.println("Nem bizonyítható szabálytalanság.");
}

System.out.println("");
```

```
                // 8. feladat
                System.out.println("A 8. feladat megoldása");
                System.out.println("Blokkolás ");
                System.out.println("Kérem a hiányzó adatokat!");

                String sikeresseg;
                int feladatkod;

                Scanner sc2 = new Scanner(System.in);

                PrintWriter kiir = new PrintWriter(new FileWriter("blokkol.txt"));

                int liftHasznalatDb = 0;
                for (i = 1; i <= igenyDb; i++) {
                        if (csapat[i - 1] == sorszam) {
                                liftHasznalatDb++;
                                System.out.println(liftHasznalatDb + ". lifthasználat");
                                kiir.println("Befejezés ideje: " + ora[i - 1] + ":" + perc[i - 1] + ":" + mp[i - 1]);
                                System.out.print("Kérem az előző munka eredményét (befejezett vagy befejezetlen): ");
                                sikeresseg = sc2.nextLine();
                                kiir.println("Sikeresség: " + sikeresseg);
                                kiir.println("------");
                                kiir.println("Indulási emelet: " + indSzint[i - 1]);
                                kiir.println("Célemelet: " + celSzint[i - 1]);
                                System.out.print("Kérem a következő munka feladatkódját (1 és 99 közötti szám): ");
                                feladatkod = sc1.nextInt();
                                kiir.println("Feladatkód: " + feladatkod);
                        }
                }

                sc1.close();
                sc2.close();
                kiir.close();

                System.out.println("");
                System.out.println("A blokkol.txt fájl kiírása befejeződött. \n");
        }

}
```

```
Az 1. feladat megoldása
Az igeny.txt fájl beolvasása

A 2. feladat megoldása
A lift kezdeti helyzetének beolvasása
Kérem az indulási szintet, ahol a lift 9 órakor áll (max. 65): 4

A 3. feladat megoldása
A lift végső helyzetének meghatározása
A lift a(z) 10. szinten áll az utolsó igény teljesítése után.

A 4. feladat megoldása
A lift szélső helyzeteinek meghatározása
A legalacsonyabb szint: 4
A legmagasabb szint: 55

Az 5. feladat megoldása
Felfelé indulások száma utassal és utas nélkül
A liftnek 57 esetben kellett felfelé menni utassal,
és 37 esetben utas nélkül.

A 6. feladat megoldása
A következő szerelőcsapatok nem vették igénybe a liftet:
6 9 17 23 25

A 7. feladat megoldása
Szabálytalankodott-e egy véletlenszerűen kiválasztott csapat?
A véletlenszerűen kiválasztott csapat sorszáma: 8
Nem bizonyítható szabálytalanság.

A 8. feladat megoldása
Blokkolás
Kérem a hiányzó adatokat!
1. lifthasználat
Kérem az előző munka eredményét (befejezett vagy befejezetlen): befejezett
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 24
2. lifthasználat
Kérem az előző munka eredményét (befejezett vagy befejezetlen): befejezetlen
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 12
3. lifthasználat
Kérem az előző munka eredményét (befejezett vagy befejezetlen): befejezett
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 44
4. lifthasználat
Kérem az előző munka eredményét (befejezett vagy befejezetlen): befejezetlen
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 2
5. lifthasználat
Kérem az előző munka eredményét (befejezett vagy befejezetlen): befejezett
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 44

A blokkol.txt fájl kiírása befejeződött.
```

```
A blokkol.txt szövegfájl

Befejezés ideje: 9:18:34
Sikeresség: befejezett
-----
Indulási emelet: 8
Célemelet: 12
Feladatkód: 24
Befejezés ideje: 10:14:45
Sikeresség: befejezetlen
-----
Indulási emelet: 12
Célemelet: 28
Feladatkód: 12
Befejezés ideje: 11:24:27
Sikeresség: befejezett
-----
Indulási emelet: 28
Célemelet: 39
Feladatkód: 44
Befejezés ideje: 12:40:43
Sikeresség: befejezetlen
-----
Indulási emelet: 39
Célemelet: 17
Feladatkód: 2
Befejezés ideje: 13:54:21
Sikeresség: befejezett
-----
Indulási emelet: 17
Célemelet: 21
Feladatkód: 44
```

2009 május idegennyelvű: Automata

A Csokibolt Kft. a város több pontján üzemeltet csokoládé-automatát. Az automatákból sokféle csokoládét lehet vásárolni pénzérmék bedobásával. A vásárláshoz az 1, 2, 5, 10, 20, 50 és 100 fabatkás érmék használhatók. Egyszerre csak egyfajta csokoládé vásárolható. A vásárlás során először ki kell választani a csokoládét, majd be kell állítani a kívánt darabszámot, végül be kell dobni a pénzt. Ha a szükségesnél több pénzt dobnak be, a gép a csokoládé mellett kiadja a visszajárót is. Amennyiben az automatában már nincs a kívánt darabszámú csokoládé, vagy a bedobott összeg nem elegendő, a vásárlás meghiúsul.

Az egyik automatában árult csokoládék lényeges adatait a *csoki.txt* állomány tartalmazza. Első sorában az automata rekeszeinek száma (legfeljebb 100) található. A második sortól kezdve soronként három szám, egy-egy rekesz adatsora olvasható. Az első szám a rekesz sorszáma, a második a rekeszben található csokoládé darabszáma, a harmadik pedig az egységára. Egy-egy rekeszben legfeljebb 100 szelet fér el, egy szelet ára legfeljebb 300 fabatka. A rekeszek sorszámozása 1-től kezdődik és folyamatos.

A vásárlások adatai a *vasarlas.txt* állományban olvashatók. Az első sorban a vásárlások száma, legfeljebb 100 olvasható. A továbbiakban soronként 9 szám szerepel, ami egy vásárlás adatait jelenti az alábbiak szerint: az első szám a választott rekesz sorszáma, a második a kívánt darabszám, utána pedig az következik, hogy az egyes címletekből hány darabot dobtak a gépbe. Az első az 1 fabatkás, a többi növekvően szerepel mögötte, így az utolsó a 100 fabatkás. Az állományban egyetlen szám sem nagyobb 100-nál.

Például:

csoki.txt

```
23
1 23 76
2 8 111
3 0 0
…
```

Az 3. sor megmutatja, hogy a 2. rekeszben 8 csokoládé van, amelynek darabja 111 fabatka.

vasarlas.txt

```
19
2 3 1 1 0 1 1 0 3
2 6 0 0 0 0 0 0 7
1 2 2 0 0 0 0 0 2
…
```

A 3. sor megmutatja, hogy a második vásárló a 2. rekeszből 6 csokoládét választott, 7 darab 100 fabatkás érmét dobott az automatába és más címletű pénzt nem.

Készítsen programot, amely az alábbi kérdésekre válaszol!
A program forráskódját *automata* néven mentse!

Minden részfeladat megoldása előtt írja a képernyőre a feladat sorszámát! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár (például a 4. feladat esetén: „`4. feladat Kérem a pénzösszeget!`”)! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be a *csoki.txt* és a *vasarlas.txt* állományban talált adatokat, s azok felhasználásával oldja meg a következő feladatokat! Ha az állományokat nem tudja beolvasni, az állományok első 8 sorának adatait jegyezze be a programba és dolgozzon azzal!
2. Milyen értékben van csokoládé az automatában? Írja képernyőre a választ a következőhöz hasonló formában: „`Az automatában 24817 fabatka értékű csokoládé van.`”!
3. Írja a képernyőre, hogy mely rekeszekből próbáltak csokoládét vásárolni! Minden rekesz sorszámát csak egyszer jelenítse meg! A számokat egymástól szóközzel elválasztva tüntesse fel!
4. Anna magának és barátainak összesen 7 egyforma csokoládét szeretne vásárolni. Kérje be a csokoládéra szánt pénzösszeget! Írja a képernyőre azon rekeszek sorszámát, amelyek közül választhat! A rekeszek sorszámát szóközökkel válassza el egymástól!
5. Okos Péter szeret mindenütt pontosan annyi pénzt átadni, amennyi a fizetendő összeg. Ezen túl szeret úgy fizetni, hogy a lehető legkevesebb pénzérmét, bankjegyet kelljen átadnia. Kérje be egy rekesz sorszámát és a darabszámot, majd írja ki, hogy a felhasznált pénzérmékből címletenként hány darabot kell bedobnia Péternek! Csak a felhasznált címleteket adja meg! Egy sorba egy címlet kerüljön; először a címlet értéke, majd mögötte a darabszám jelenjen meg! Nem kell vizsgálnia, hogy van-e elég csokoládé a rekeszben! A megoldás során segítségként a következő algoritmust használhatja: *Keresse meg a legnagyobb címletet, amely nem haladja meg a fizetendő összeget! Ebből a címletből kell egyet használnia! A fizetendőt csökkentse a címlet értékével, majd kezdje elölről az algoritmust, ha az nem nulla!* Ez az algoritmus a feladatban szereplő címletek esetén működik, de létezhet olyan címletlista, amelynél nem alkalmazható.
6. Írja a *rekesz7.txt* állományba, hogy hányas sorszámú vásárlások során hány darabot vettek a 7-es rekeszből! Vegye figyelembe, hogy minden sikeres vásárlással csökken a rekeszben lévő csokoládék száma! Soronként egy vásárlási próbálkozást tüntessen fel! A sor elején a vásárlási próbálkozás sorszáma jelenjen meg, tőle tabulátorral (ASCII kódja a 9-es) elválasztva pedig a vásárlás eredménye legyen olvasható! Az eredmény sikeres vásárlás esetén a darabszám. Ha nem volt megadott mennyiségnek megfelelő csokoládé, akkor a sorszám mögé a „`ke vés a csoki`” üzenet kerüljön! Ha a vásárló által bedobott pénzösszeg kevés, akkor a „`nem volt elég pénz`” szöveget írja a fájlba! Amennyiben a vásárlás több okból is meghiúsulhat, elegendő csak az egyik okot megjeleníteni.

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Automata
 *
 * @author Klemand
 */

public class EmeltInfo2009mid {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A csoki.txt és a vasarlas.txt fájlok beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("csoki.txt"));

		String sor = beolvas.readLine();

		int rekeszDb = Integer.parseInt(sor);

		int[][] csoki = new int[rekeszDb][3];
		// rekesz, darabszám, egységár

		int i, j;
		String[] daraboltSor;
		for (i = 1; i <= rekeszDb; i++) {
			daraboltSor = beolvas.readLine().split(" ");
			for (j = 1; j <= 3; j++) {
				csoki[i - 1][j - 1] = Integer.parseInt(daraboltSor[j - 1]);
			}
		}
		beolvas.close();

		beolvas = new BufferedReader(new FileReader("vasarlas.txt"));

		sor = beolvas.readLine();

		int vasarlasDb = Integer.parseInt(sor);

		int[][] vasarlas = new int[vasarlasDb][9];
		// rekesz, darabszám, bedobott 1, 2, 5, 10, 20, 50 és 100 fabatkások száma

		for (i = 1; i <= vasarlasDb; i++) {
			daraboltSor = beolvas.readLine().split(" ");
			for (j = 1; j <= 9; j++) {
				vasarlas[i - 1][j - 1] = Integer.parseInt(daraboltSor[j - 1]);
			}
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Milyen értékben van csokoládé az automatában?");

		int osszertek = 0;
		for (i = 1; i <= rekeszDb; i++) {
			osszertek += csoki[i - 1][1] * csoki[i - 1][2];
		}

		System.out.println("Az automatában " + osszertek + " fabatka értékű csokoládé van. \n");
```

```
System.out.println("A 3. feladat megoldása");
System.out.println("A következő rekeszekből próbáltak vásárolni: ");

boolean rekesz[] = new boolean[rekeszDb];

for (i = 1; i <= rekeszDb; i++) { // Rekesztörlés
        rekesz[i - 1] = false;
}

// Volt-e vásárlás az egyes rekeszekből?
int aktRekesz;
for (i = 1; i <= vasarlasDb; i++) {
        aktRekesz = vasarlas[i - 1][0];
        rekesz[aktRekesz - 1] = true;
}

// Az igazra állított rekeszek kiírása
for (i = 1; i <= rekeszDb; i++) {
        if (rekesz[i - 1]) {
                System.out.print(i + " ");
        }
}
System.out.println("\n");

System.out.println("A 4. feladat megoldása");
System.out.println("Anna lehetőségei ");

System.out.println("Kedves Anna, hány fabatkát szánsz a 7 egyforma csokoládéra? ");
System.out.print("(Egy csokoládé ára max. 300 fabatka.): ");

Scanner sc = new Scanner(System.in);
int keret = sc.nextInt();

System.out.println("");
System.out.println("A következő rekeszekből választhatsz: ");

for (i = 1; i <= rekeszDb; i++) {
        if (csoki[i - 1][1] >= 7 && 7 * csoki[i - 1][2] <= keret) {
                System.out.print(csoki[i - 1][0] + " ");
        }
}
System.out.println("\n");

System.out.println("Az 5. feladat megoldása");
System.out.println("Okos Péter címletezése  ");

System.out.print("Kérem a rekesz sorszámát: ");
int sorszam = sc.nextInt();
System.out.print("Kérem a vásárolni kívánt csokik számát: ");
int darab = sc.nextInt();

int[] cimlet = { 1, 2, 5, 10, 20, 50, 100 };
int[] cimletDb = { 0, 0, 0, 0, 0, 0, 0 };
// Tömb megadása a kezdőértékek felsorolásával

int fizetendo = csoki[sorszam - 1][2] * darab;
System.out.println("A fizetendő összeg: " + fizetendo + " fabatka.");
```

```
                do {
                        i = 7;
                        while (i >= 1 && (cimlet[i - 1] > fizetendo)) {
                                i--;
                        }
                        cimletDb[i - 1]++;
                        fizetendo -= cimlet[i - 1];
                } while (fizetendo > 0);

                System.out.println("A szükséges érmék és darabszámuk: ");
                for (i = 7; i >= 1; i--) {
                        if (cimletDb[i - 1] > 0) {
                                System.out.println(cimlet[i - 1] + " " + cimletDb[i - 1]);
                        }
                }

                System.out.println("\nA 6. feladat megoldása tetszőleges rekesz esetén");
                System.out.println("A bekért (a feladatban a 7.) rekesz ");
                System.out.println("vásárlási naplójának fájlba írása ");

                System.out.print("Kérem a rekesz sorszámát: ");
                sorszam = sc.nextInt();

                PrintWriter kiir = new PrintWriter(new FileWriter("rekesz" + sorszam + ".txt"));

                int ar, bedobott;
                for (i = 1; i <= vasarlasDb; i++) {
                        if (vasarlas[i - 1][0] == sorszam) {
                                if (vasarlas[i - 1][1] > csoki[sorszam - 1][1]) {
                                        kiir.println(i  + "\t" + "kevés a csoki");
                                } else {
                                        ar = vasarlas[i - 1][1] * csoki[sorszam - 1][2];
                                        bedobott = 0;
                                        for (j = 1; j <= 7; j++) {
                                                bedobott += vasarlas[i - 1][j + 1] * cimlet[j - 1];
                                        }
                                        if (ar > bedobott) {
                                                kiir.println(i + "\t" + "nem volt elég pénz");
                                        } else {
                                                kiir.println(i + "\t" + vasarlas[i - 1][1]);
                                                csoki[sorszam - 1][1] -= vasarlas[i - 1][1];
                                        }
                                }
                        }
                }

                sc.close();
                kiir.close();
                System.out.println("A rekesz" + sorszam + ".txt fájl kiírása befejeződött. \n");
        }

}
```

```
Az 1. feladat megoldása
A csoki.txt és a vasarlas.txt fájlok beolvasása
A beolvasás megtörtént.
A 2. feladat megoldása
Milyen értékben van csokoládé az automatában?
Az automatában 24817 fabatka értékű csokoládé van.

A 3. feladat megoldása
A következő rekeszekből próbáltak vásárolni:
1 3 4 5 7 8 9 11 12 13 15 16 17 18 19 20 22 23 24 25

A 4. feladat megoldása
Anna lehetőségei
Kedves Anna, hány fabatkát szánsz a 7 egyforma csokoládéra?
(Egy csokoládé ára max. 300 fabatka.): 750

A következő rekeszekből választhatsz:
1 3 9 10 12 13 16 17 20 21 24

Az 5. feladat megoldása
Okos Péter címletezése
Kérem a rekesz sorszámát: 13
Kérem a vásárolni kívánt csokik számát: 17
A fizetendő összeg: 1547 fabatka.
A szükséges érmék és darabszámuk:
100 15
20 2
5 1
2 1

A 6. feladat megoldása tetszőleges rekesz esetén
A bekért (a feladatban a 7.) rekesz
vásárlási naplójának fájlba írása
Kérem a rekesz sorszámát: 7
A rekesz7.txt fájl kiírása befejeződött.


A rekesz7.txt szövegfájl

2       1
9       1
21      kevés a csoki
46      1
47      kevés a csoki
55      kevés a csoki
```

2009 október: Útépítés

Az Alsó és Felső várost összekötő út 1 000 m hosszú részének a felújításán dolgoznak. Ennek a szakasznak a forgalmát figyeljük egy nap néhány óráján keresztül. Az említett szakaszon előzési tilalom van érvényben.

A forgalmat a *forgalom.txt* állomány tartalmazza. Első sorában a megfigyelési időszakban áthaladó járművek száma (legfeljebb 2000) látható, a továbbiakban pedig soronként egy áthaladó jármű adatai olvashatók időrendben. Egy sorban az első három szám azt az időpontot jelöli (óra, perc, másodperc), amikor a jármű belép a vizsgált útszakaszra. A következő szám jelöli, hogy a jármű az érintett távolságot hány másodperc alatt tenné meg (legfeljebb 300) – a belépéskor mért sebességgel –, ha haladását semmi nem akadályozná. Ezt egy betű követi, amely jelzi, hogy a jármű melyik város irányából érkezett. Ennek megfelelően a betű A vagy F lehet. Az egyes adatokat pontosan egy szóköz választja el egymástól.

Ha az útszakaszon egyik jármű utoléri a másikat, akkor az előzési tilalom miatt úgy tekintjük, hogy változatlan sorrendben, ugyanabban az időpillanatban hagyják el a szakasz, mint ahogy a lassabb jármű tenné.

Például:

forgalom.txt

```
1105
7 21 1 60 F
7 21 58 69 F
7 22 4 117 F
7 22 39 155 A
7 23 11 99 A
…
```

A 3. sor megmutatja, hogy a 7 óra 21 perc 58 másodperckor a Felső város felől érkező jármű 69 másodperc alatt tenné meg ezt az 1 km hosszú távolságot. Ez a jármű – ha más járművek nem akadályozzák – 7 óra 23 perc 7 másodperckor lép ki az útszakaszról, tehát akkor már nem tartózkodik ott.

Készítsen programot, amely az alábbi kérdésekre válaszol! A program forráskódját *ut* néven mentse! Ügyeljen arra, hogy programjának más bemeneti állomány esetén is működnie kell!

Minden részfeladat megoldása előtt írja a képernyőre annak sorszámát! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár (például a 2. feladat esetén: „`2. feladat Adja meg a jármű sorszámát!`”)!

1. Olvassa be a *forgalom.txt* állományban talált adatokat, s azok felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, akkor az első 10 sorának adatait jegyezze be a programba és dolgozzon azzal!
2. Írja ki a képernyőre, hogy az n-edikként belépő jármű melyik város felé haladt! Ehhez kérje be a felhasználótól az n értékét!
3. Írja a képernyőre, hogy a Felső város irányába tartó utolsó két jármű hány másodperc különbséggel érte el az útszakasz kezdetét!

4. Határozza meg óránként és irányonként, hogy hány jármű érte el a szakaszt! Soronként egy-egy óra adatait írja a képernyőre! Az első érték az órát, a második érték az Alsó, a harmadik a Felső város felől érkező járművek számát jelentse! A kiírásban csak azokat az órákat jelenítse meg, amelyekben volt forgalom valamely irányban!
5. A belépéskor mért értékek alapján határozza meg a 10 leggyorsabb járművet! Írassa ki a képernyőre ezek belépési idejét, a várost (`Alsó`, illetve `Felső`), amely felől érkezett, és m/s egységben kifejezett sebességét egy tizedes pontossággal, sebességük szerinti csökkenő sorrendben! Ha több azonos sebességű járművet talál, bármelyiket megjelenítheti. Soronként egy jármű adatait jelenítse meg, és az egyes adatokat szóközzel tagolja! (A feladat megoldásakor figyeljen arra, hogy a következő feladatban az adatok eredeti sorrendjét még fel kell használni!)
6. Írassa ki az *`also.txt`* állományba azokat az időpontokat, amikor az Alsó város felé tartók elhagyták a kérdéses útszakaszt! Ha egy jármű utolér egy másikat, akkor a kilépésük időpontja a lassabb kilépési ideje legyen! A fájl minden sorába egy-egy időpont kerüljön óra perc másodperc formában! A számokat pontosan egy szóköz válassza el egymástól!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Útépítés
 *
 * @author Klemand
 */

public class EmeltInfo2009okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása a forgalom.txt fájlból");

		BufferedReader beolvas = new BufferedReader(new FileReader("forgalom.txt"));

		String sor;
		sor = beolvas.readLine();

		int n = Integer.parseInt(sor);
		int[] ora = new int[n]; // érkezési idő
		int[] perc = new int[n];
		int[] mp = new int[n];
		int[] minIdo = new int[n]; // minimálisan szükséges áthaladási idő
		String[] honnan = new String[n];

		String[] daraboltSor;

		for (int i = 1; i <= n; i++) {
			daraboltSor = beolvas.readLine().split(" ");
			ora[i - 1] = Integer.parseInt(daraboltSor[0]);
			perc[i - 1] = Integer.parseInt(daraboltSor[1]);
			mp[i - 1] = Integer.parseInt(daraboltSor[2]);
			minIdo[i - 1] = Integer.parseInt(daraboltSor[3]);
			honnan[i - 1] = (daraboltSor[4]);
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Merre haladt a beolvasott sorszámú jármű?");

		Scanner sc = new Scanner(System.in);
		System.out.print("Kérem a jármű sorszámát!  ");
		int sorszam = sc.nextInt();
		sc.close();

		System.out.print("A(z) " + sorszam + ". autó ");
		System.out.println(iranyMeghat(honnan[sorszam - 1]) + " város felé haladt. \n");

		System.out.println("A 3. feladat megoldása");
		System.out.print("A Felső város irányába haladó utolsó két jármű ");

		int[] erkezes = new int[n];
		int i;
		for (i = 1; i <= n; i++) {
			erkezes[i - 1] = 3600 * ora[i - 1] + 60 * perc[i - 1] + mp[i - 1];
		}
```

```
i = n;
int k = 3;
// k számlál és előjelez is

int idokulonbseg = 0;

while (i >= 1 && k > -1) {
        if ((honnan[i - 1].equals("A"))) {
                k -= 2; // az utolsónál 1 lesz az értéke, az utolsó előttinél -1
                idokulonbseg += k * erkezes[i - 1];
        }
        i--;
}

if (i >= 1) {
        System.out.println(idokulonbseg + " s időkülönbséggel érte el az útszakasz kezdetét.\n");
} else {
        System.out.println("Nem haladt két jármű Felső város irányába.\n");
}

System.out.println("A 4. feladat megoldása");
System.out.println("Az óránként Alsó, ill. Felső város felől érkező autók száma: ");

int[] dbA = new int[24];
int[] dbF = new int[24];

for (i = 0; i <= 23; i++) { // A számlálók nullázása
        dbA[i] = 0;
        dbF[i] = 0;
}

for (i = 1; i <= n; i++) {
        if (honnan[i - 1].equals("A")) {
                dbA[ora[i - 1]]++;
        } else {
                dbF[ora[i - 1]]++;
        }
}

for (i = 0; i <= 23; i++) {
        if (dbA[i] > 0 || dbF[i] > 0) {
                System.out.println(i + " " + dbA[i] + " " + dbF[i]);
        }
}

System.out.println("\nAz 5. feladat megoldása");
System.out.println("A 10 leggyorsabb autó ");

int leglassabb = 1;
for (i = 2; i <= n; i++) {
        if (minIdo[i - 1] > minIdo[leglassabb - 1]) {
                leglassabb = i;
        }
}
/*
 * Megjelöljük a gyorsak közé kiválasztott autókat. A leglassabb lesz minden
 * ciklus elején az aktuális leggyorsabb, így nem kell vizsgálni, hogy
 * kiválasztottuk-e már a gyorsak közé.
 */

boolean[] gyors = new boolean[n];

for (i = 1; i <= n; i++) {
        gyors[i - 1] = false; // kezdőérték megadása
}
```

```
                double sebesseg = 0;
                int j;
                for (i = 1; i <= 10; i++) {
                        int leggyorsabb = leglassabb;
                        for (j = 1; j <= n; j++) {
                                if ((minIdo[j - 1] < minIdo[leggyorsabb - 1]) && !gyors[j - 1]) {
                                        leggyorsabb = j;
                                }
                        }
                        gyors[leggyorsabb - 1] = true;
                        sebesseg = (double) 1000 / minIdo[leggyorsabb - 1];
                        System.out.print(ora[leggyorsabb - 1] + " " + perc[leggyorsabb - 1]);
                        System.out.print(" " + mp[leggyorsabb - 1]);
                        System.out.print(" " + nevMeghat(honnan[leggyorsabb - 1]));
                        System.out.printf(" %.1f", sebesseg);
                        System.out.println(" m/s");
                }

                System.out.println("\nA 6. feladat megoldása");
                System.out.println("Az Alsó város felé tartó autók kilépési időinek kiíratása az also.txt fájlba ");

                PrintWriter kiir = new PrintWriter(new FileWriter("also.txt"));

                int kilepes = -1;

                for (i = 1; i <= n; i++) {
                        if (honnan[i - 1].equals("F")) {
                                if (erkezes[i - 1] + minIdo[i - 1] > kilepes) {
                                        kilepes = erkezes[i - 1] + minIdo[i - 1];
                                }
                                kiir.println(idoKonvertalas(kilepes));
                        }
                }

                kiir.close();
                System.out.println("A fájlkiírás befejeződött. \n");

        }

        public static String iranyMeghat(String honnan) {
                if (honnan.equals("A")) {
                        return "Felső";
                } else {
                        return "Alsó";
                }
        }

        public static String nevMeghat(String honnanBe) {
                if (honnanBe.equals("A")) {
                        return "Alsó";
                } else {
                        return "Felső";
                }
        }

        public static String idoKonvertalas(int ido) {
                int ora, perc, mp;
                mp = ido % 60;
                perc = ido / 60;
                ora = perc / 60;
                perc = perc % 60;
                return "" + ora + " " + perc + " " + mp; // visszaadott függvényérték
        }

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása a forgalom.txt fájlból
A beolvasás megtörtént.

A 2. feladat megoldása
Merre haladt a beolvasott sorszámú jármű?
Kérem a jármű sorszámát!  100
A(z) 100. autó Alsó város felé haladt.

A 3. feladat megoldása
A Felső város irányába haladó utolsó két jármű 228 s időkülönbséggel érte el az útszakasz kezdetét.

A 4. feladat megoldása
Az óránként Alsó, ill. Felső város felől érkező autók száma:
7 39 44
8 58 65
9 57 66
10 59 50
11 55 61
12 59 62
13 74 53
14 51 78
15 60 61
16 30 23

Az 5. feladat megoldása
A 10 leggyorsabb autó
10 38 23 Alsó 25,0 m/s
12 54 34 Alsó 25,0 m/s
13 27 12 Alsó 25,0 m/s
13 42 22 Alsó 25,0 m/s
16 0 1 Alsó 25,0 m/s
7 28 18 Felső 24,4 m/s
9 4 41 Alsó 24,4 m/s
9 51 46 Felső 24,4 m/s
10 16 36 Alsó 24,4 m/s
12 25 33 Felső 24,4 m/s

A 6. feladat megoldása
Az Alsó város felé tartó autók kilépési időinek kiíratása az also.txt fájlba
A fájlkiírás befejeződött.
```

Az also.txt szövegfájl

```
7 22 1
7 23 7
7 24 1
7 26 15
7 26 20
7 27 7
7 28 9
7 28 20
7 28 40
7 29 46
7 29 46
7 31 21
7 32 37
7 35 52
7 36 39
7 37 25
7 39 25
7 42 20
7 42 20
7 42 20
7 43 42
...
```

2010 május: Helyjegy

Egy autóbuszokat üzemeltető társaság távolsági járataira az utasok jobb kiszolgálása érdekében csak akkor ad el jegyet, ha ülőhelyet is tud biztosítani. Minden jegyre rányomtatja, hogy az adott vonalon mettől meddig érvényes és melyik ülést lehet elfoglalni birtokában.

Az *eladott.txt* állomány pontosan egy út jegyvásárlásait tartalmazza. Az első sorban az eladott jegyek száma (legfeljebb 500), a vonal hossza (legfeljebb 200 km) és minden megkezdett 10 km után fizetendő összeg (legfeljebb 100 Ft) található.

Az állomány további sorai — a vásárlás sorrendjében — egy-egy jegy három adatát írják le: az utas melyik ülést foglalhatja el, hol száll fel és hol száll le. (A fel- és a leszállás helyét a járat kezdőállomásától mért távolsággal adják meg.) Az üléseket 1-től 48-ig folyamatosan számozták. A soron belüli határoló jel minden esetben egy-egy szóköz. Az állomány csak egész számokat tartalmaz.

Az utast a későbbiekben egyetlen sorszámmal azonosítjuk, azzal az értékkel, amely megadja, hogy hanyadik jegyvásárló volt.

A jegy árának meghatározásakor az értéket öttel osztható számra kell kerekítenie. (1, 2, 6 és 7 esetén lefelé, 3, 4, 8 és 9 esetén pedig felfelé kell kerekítenie.)

Például:

eladott.txt

```
132 200 71
20 0 110
12 13 65
…
```

Az adott járat 200 km hosszú úton közlekedik. Eddig 132 jegyet adtak el, és megkezdett 10 km-ként 71 Ft-ba kerül a jegy. Az állomány harmadik sora tartalmazza a második jegyvásárló adatait, aki a 13. és a 65. km között utazik a 12. helyen ülve. A megtett távolság 52 km, tehát 6 darab 10 km hosszú szakaszért kell fizetnie, ennek értéke 6*71, azaz 426 Ft. Mivel kerekíteni kell, ezért a fizetendő összeg 425 Ft.

Készítsen programot, amely az alábbi kérdésekre válaszol! A program forráskódját *helyjegy* néven mentse!

Minden – képernyőre írást igénylő – részfeladat megoldása előtt írja a képernyőre a feladat sorszámát! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár (például a 7. feladat esetén: „`7. feladat Adja meg, hogy az út mely kilométerén kéri az utaslistát!`”)! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be az *eladott.txt* állományban talált adatokat, s azok felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, az állomány első 10 sorának adatait jegyezze be a programba és dolgozzon azzal!
2. Adja meg a legutolsó jegyvásárló ülésének sorszámát és az általa beutazott távolságot! A kívánt adatokat a képernyőn jelenítse meg!
3. Listázza ki, kik utazták végig a teljes utat! Az utasok sorszámát egy-egy szóközzel elválasztva írja a képernyőre!

4. Határozza meg, hogy a jegyekből mennyi bevétele származott a társaságnak! Az eredményt írja a képernyőre!
5. Írja a képernyőre, hogy a busz végállomást megelőző utolsó megállásánál hányan szálltak fel és le!
6. Adja meg, hogy hány helyen állt meg a busz a kiinduló állomás és a célállomás között! Az eredményt írja a képernyőre!
7. Készítsen „utaslistát” az út egy pontjáról! A listában ülésenként tüntesse fel, hogy azt az adott pillanatban melyik utas foglalja el! A pontot, azaz a kiindulási állomástól mért távolságot, a felhasználótól kérje be! Ha a beolvasott helyen éppen megálló lett volna, akkor a felszálló utasokat vegye figyelembe, a leszállókat pedig hagyja figyelmen kívül!
   Az eredményt az ülések sorszámának sorrendjében írja a *kihol.txt* állományba!
   Az üres helyek esetén az „`üres`” szót jelenítse meg! Minden ülés külön sorba kerüljön!

Például:

kihol.txt

```
1. ülés: üres
2. ülés: üres
3. ülés: üres
4. ülés: 29. utas
5. ülés: 95. utas
…
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Helyjegy
 *
 * @author Klemand
 */

public class EmeltInfo2010maj {

    public static void main(String[] args) throws IOException {

        System.out.println("Az 1. feladat megoldása");
        System.out.println("Az adatok beolvasása az eladott.txt fájlból \n");

        BufferedReader beolvas = new BufferedReader(new FileReader("eladott.txt"));

        String sor;
        sor = beolvas.readLine();

        String[] daraboltSor = sor.split(" ");

        int n = Integer.parseInt(daraboltSor[0]);//utasok száma
        int tav = Integer.parseInt(daraboltSor[1]);
        int egysAr = Integer.parseInt(daraboltSor[2]);

        int[][] busz = new int[n][3];
        int i;
        for (i = 1; i <= n; i++) {
            daraboltSor = beolvas.readLine().split(" ");
            busz[i - 1][0] = Integer.parseInt(daraboltSor[0]);//hely
            busz[i - 1][1] = Integer.parseInt(daraboltSor[1]);//felszállás
            busz[i - 1][2] = Integer.parseInt(daraboltSor[2]);//leszállás
        }

        beolvas.close();

        System.out.println("A 2. feladat megoldása");
        System.out.println("Az utolsó felszálló adatai: ");
        System.out.println("Ülésének száma: " + busz[n - 1][0] + ", beutazott távolság: "
                           + (busz[n - 1][2] - busz[n - 1][1]) + " km");
        System.out.println("");

        System.out.println("A 3. feladat megoldása");
        System.out.print("A teljes távot végigutazók: ");

        for (i = 1; i <= n; i++) {
            if (busz[i - 1][1] == 0 && busz[i - 1][2] == tav) {
                System.out.print(i + " ");
            }
        }
        System.out.println("\n");
```

```
System.out.println("A 4. feladat megoldása \n");
System.out.print("A társaság bevétele: ");

int bevetel = 0;

for (i = 1; i <= n; i++) {
    int jegyar = jegyarMeghat(busz[i - 1][1], busz[i - 1][2], egysAr);
    bevetel += jegyar;
}
System.out.println(bevetel + " Ft");
System.out.println("");

System.out.println("Az 5. feladat megoldása");

System.out.print("A végállomás előtti utolsó megálló ");

int[][] km = new int[tav + 1][2];

for (i = 0; i <= tav; i++) { //A távolság 0-val kezdődik!
    km[i][0] = 0; //felszállók száma km-enként
    km[i][1] = 0; //leszállók száma km-enként
}

int kmFel, kmLe;
for (i = 1; i <= n; i++) {
kmFel=busz[i - 1][1];
kmLe=busz[i - 1][2];
    km[kmFel][0]++;
    km[kmLe][1]++;
}
/* A felszállás km-énél növelem a felszállók számát,
a leszállás km-énél pedig a leszállók számát*/

i = tav - 1;
while (i >= 0 && (km[i][0] + km[i][1] == 0)) {
    i--;
}

if (i >= 0) {
    System.out.println("a(z) " + i + " km-nél volt.");
    System.out.println(km[i][0] + " felszálló és " + km[i][1] + " leszálló volt.");
} else {
    System.out.println("A busz végig üresen utazott.");
}

System.out.println("");

System.out.println("A 6. feladat megoldása");
System.out.print("A busz a kiinduló állomás és a célállomás között ");

int megallo = 0;

for (i = 1; i < tav; i++) {
    if (km[i][0] + km[i][1] > 0) {
        megallo++;
    }
}

System.out.println(megallo + " helyen állt meg. \n");
```

```
        System.out.println("A 7. feladat megoldása");
        System.out.println("Utaslista egy megadott távolságnál  \n ");
        System.out.print("Kérem a kiindulási helytől való távolságot km-ben (max. " +tav +"): ");

        Scanner sc = new Scanner(System.in);

        int tavolsag = sc.nextInt();

        int ulesDb = 48; //Az ülések száma
        int[] ules = new int[ulesDb];

        for (i = 1; i <= ulesDb; i++) {
            ules[i - 1] = 0;
        }

        int aktUles;

        for (i = 1; i <= n; i++) {
            if (tavolsag >= busz[i - 1][1] && tavolsag < busz[i - 1][2]) {
                //Az i-edik utas a buszon tartózkodik
                aktUles = busz[i - 1][0];
                ules[aktUles - 1] = i;
            }
        }

        PrintWriter kiir = new PrintWriter(new FileWriter("kihol.txt"));

        for (i = 1; i <= ulesDb; i++) {
            kiir.print(i + ". ülés: ");
            if (ules[i - 1] > 0) {
                kiir.println(ules[i - 1] + ". utas");
            } else {
                kiir.println("üres");
            }
        }

        sc.close();
        kiir.close();

        System.out.println("");
        System.out.println("A fájlkiírás befejeződött. \n");
    }

    public static int jegyarMeghat(int fel, int le, int egysAr) {

        int tav= le-fel;
        int tavEgys = tav/10;

        if (tav % 10 > 0) {
            tavEgys += 1;
        }

        int jegyar = tavEgys * egysAr;
        int maradek = jegyar % 5;

        jegyar -= maradek;

        if (maradek >= 3) {
            jegyar += 5;
        }

        return jegyar;
    }

}
```

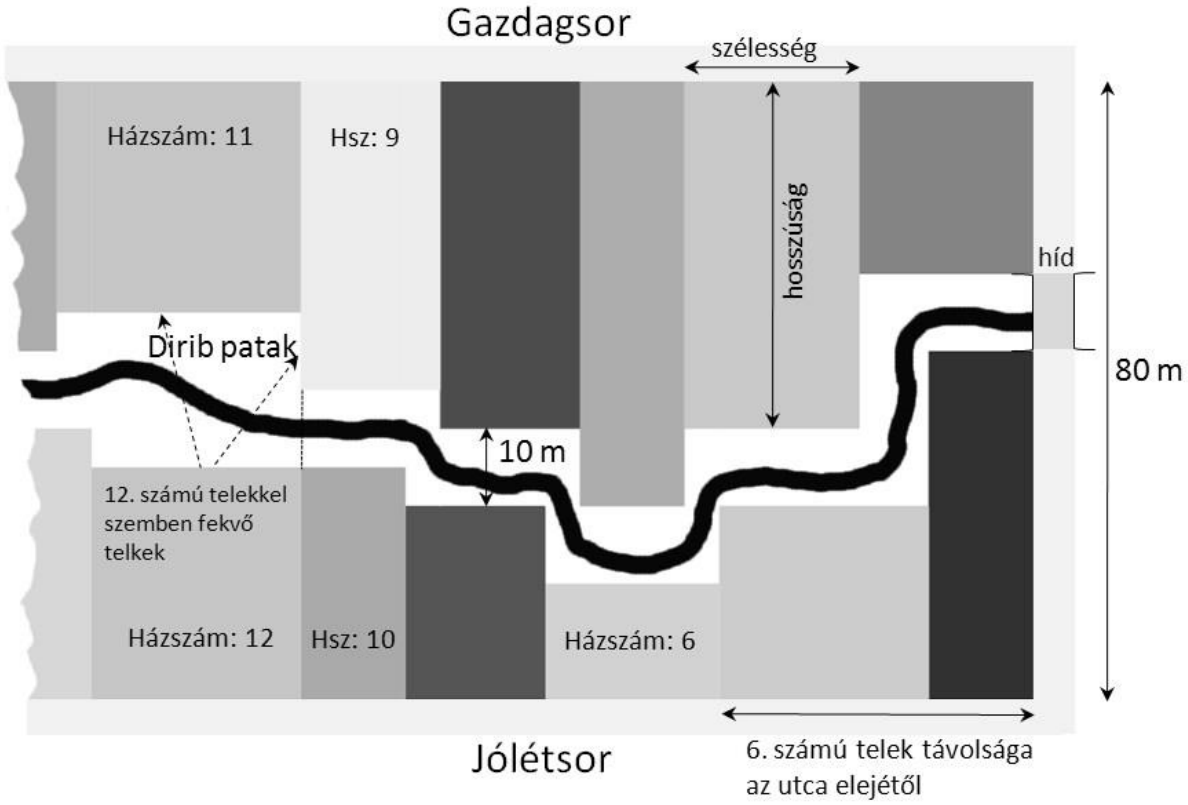
```
Az 1. feladat megoldása
Az adatok beolvasása az eladott.txt fájlból

A 2. feladat megoldása
Az utolsó felszálló adatai:
Ülésének száma: 29, beutazott távolság: 50 km

A 3. feladat megoldása
A teljes távot végigutazók: 67 71 95

A 4. feladat megoldása

A társaság bevétele: 46210 Ft

Az 5. feladat megoldása
A végállomás előtti utolsó megálló a(z) 165 km-nél volt.
2 felszálló és 5 leszálló volt.

A 6. feladat megoldása
A busz a kiinduló állomás és a célállomás között 24 helyen állt meg.

A 7. feladat megoldása
Utaslista egy megadott távolságnál

Kérem a kiindulási helytől való távolságot km-ben (max. 172): 56

A fájlkiírás befejeződött.
```

A kihol.txt szövegfájl

```
1. ülés: 42. utas
2. ülés: 58. utas
3. ülés: 60. utas
4. ülés: 33. utas
5. ülés: 95. utas
6. ülés: 63. utas
7. ülés: 46. utas
8. ülés: 96. utas
9. ülés: üres
10. ülés: 41. utas
11. ülés: üres
12. ülés: 102. utas
13. ülés: 9. utas
14. ülés: 71. utas
15. ülés: 22. utas
16. ülés: 67. utas
17. ülés: 104. utas
18. ülés: 61. utas
19. ülés: 40. utas
20. ülés: 26. utas
21. ülés: 109. utas
22. ülés: üres
23. ülés: 45. utas
24. ülés: 97. utas
25. ülés: 92. utas
26. ülés: 24. utas
27. ülés: 37. utas
28. ülés: 64. utas
29. ülés: üres
30. ülés: 12. utas
31. ülés: 112. utas
32. ülés: 73. utas
33. ülés: 91. utas
34. ülés: 68. utas
35. ülés: 70. utas
36. ülés: 57. utas
37. ülés: 54. utas
38. ülés: 43. utas
39. ülés: üres
40. ülés: 7. utas
41. ülés: 90. utas
42. ülés: 85. utas
43. ülés: 32. utas
44. ülés: 47. utas
45. ülés: 19. utas
46. ülés: üres
47. ülés: 82. utas
48. ülés: 101. utas
```

2010 május idegennyelvű: Telek

Patakfalván a faluszélen levő beépítetlen területet szeli ketté a Dirib patak. Az önkormányzat elhatározta, hogy építési telkek kialakításával létrehozza a Szép jövő lakótelepet. A beépítés után egy téglalap alakú területen két utca jön létre: Gazdagsor és Jólétsor. A két sor lakói „lábszomszédok”, de telkeiket elválasztja egymástól a Dirib patak. A két utca párhuzamos, az utcafrontokat 80 méter választja el egymástól. Mindkét soron azonos számú téglalap alakú telket jelöltek ki, soronként legfeljebb 30-at. Gazdagsoron csak páratlan, Jólétsoron csak páros házszámokat adnak ki (1-től, illetve 2-től indulva kihagyásmentesen számozva). Egy telek szélessége maximum 40 méter. Az utcák végén egy-egy híd köti össze a patak két partját. A telkek kijelölésénél figyelembe vették a patak medrének nyomvonalát.



A kijelölt telkekről kimutatás készült, amit a `telkek.txt` fájl tartalmaz. Ennek a fájlnak az első sora tartalmazza a kiosztandó telkek számát, majd az ezt követő sorokban az egyes telkek adatai találhatók.
Az első adat a házszám, a második a telek szélessége, míg a harmadik az erre merőlegesen mért hosszúsága. Gazdagsor esetén az összes adat rendelkezésre áll, Jólétsor esetében viszont a hosszúság adatok helyén 0 áll. Az adatok között pontosan egy szóköz található.

Készítsen programot `telek` néven, amely az alábbi kérdésekre válaszol!

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát! (Például `3. feladat:`)

1. Olvassa be a *telkek.txt* állományban található adatokat, s annak felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, akkor a benne található adatok közül Gazdagsor 1., 3., 5., 7. és 9. számú, valamint Jólétsor 2., 4., 6., 8. és 10. számú telkének adatait jegyezze be a programba, s úgy oldja meg a feladatokat!
2. Hány métert kell annak gyalogolnia, aki körbe akarja járni a két utcát? A kiszámított távolságot írassa ki a képernyőre!
3. Az önkormányzat előírásai szerint a 20 m széles vagy annál keskenyebb telkek esetén teljes utcafront beépítést kell alkalmazni. Határozza meg és a képernyőre írassa ki, hogy ez hány telekre vonatkozik a Jólétsoron!
4. Hány háznyira van egymástól a legnagyobb és a legkisebb területű telek Gazdagsoron? A két telek között elhelyezkedő telkek számát, valamint a legnagyobb és legkisebb telek házszámát, illetve területét írassa ki a képernyőre!
5. Az önkormányzat telekadót fog kivetni. Az adót Fabatkában számolják. A 700 négyzetméteres és annál kisebb telkek esetén ez 51 Fabatka négyzetméterenként, az ennél nagyobb telkeknél az első 700 négyzetméterre vonatkozóan szintén 51 Fabatka, 700 négyzetméter felett egészen 1000 négyzetméterig 39 Fabatka a négyzetméterenkénti adó. Az 1000 négyzetméter feletti részért négyzetméter árat nem, csak 200 Fabatka egyösszegű általányt kell fizetni. A 15 m vagy annál keskenyebb, illetve a 25 m vagy annál rövidebb telkek tulajdonosai 20% adókedvezményben részesülnek. Az adó meghatározásánál 100 Fabatkás kerekítést kell használni (pl. 6238 esetén 6200, 6586 esetén 6600). Határozza meg, mekkora adóbevételre számíthat Gazdagsor után az önkormányzat!
6. Melyik a 3 utolsó telek a Jólétsoron? A házszámokat és a telkeknek a Jólétsor elejétől mért távolságát írja ki a képernyőre a házszámok szerint csökkenő sorrendben!
7. Határozza meg Jólétsor telkeinek hosszúságát! Vegye figyelembe, hogy a szemben fekvő telkek patak felőli határvonalait az utcafrontra merőleges irányban legalább 10 méternek kell elválasztania egymástól! Szemben fekvőnek számítanak a telkek akkor is, ha csak a telkek valamelyik széle van egymással szemben. (Például a 10-es számú telekkel csak a 9es és 11-es számú telek van szemben.) A számításnál a feltételnek megfelelő legnagyobb telkeket kell kialakítani! Jólétsor adatait írja ki a *joletsor.csv* fájlba! Az egyes sorokban a házszám, a szélesség és a hosszúság szerepeljen! Az adatokat pontosan egy pontosvessző válassza el egymástól!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * Telek
 *
 * @author Klemand
 */

public class EmeltInfo2010mid {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása az telkek.txt fájlból");

		BufferedReader beolvas = new BufferedReader(new FileReader("telkek.txt"));

		String sor = beolvas.readLine();

		int n = Integer.parseInt(sor);
		//n páros, mert ugyanannyi ház van a két soron

		String[] daraboltSor;
		int[][] telek = new int[n][4];
		/*
		 * házszám, szélesség, hossz,
		 * és tárolni fogjuk az utcák elejétől mért távolságot is
		 * házszám: Jólétsor páros, Gazdagsor páratlan
		 */

		int i, j;
		for (i = 1; i <= n; i++) {
			daraboltSor = beolvas.readLine().split(" ");
			for (j = 1; j <= 3; j++) {
				telek[i - 1][j - 1] = Integer.parseInt(daraboltSor[j - 1]);
			}
		}
		beolvas.close();

		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.print("Egy körséta hossza: ");
		int oldal = 80;

		// Házszámok szerinti rendezés

		int[] asztal = new int[4];

		for (i = n - 1; i >= 1; i--) {
			for (j = 1; j <= i; j++) {
				if (telek[j - 1][0] > telek[j][0]) {
					asztal = telek[j - 1];
					telek[j - 1] = telek[j];
					telek[j] = asztal;
				}
			}
		}
```

```
// A telkek kezdetének távolsága az utcák elejétől
telek[1][3] = 0; //páros házszámok (Jólétsor)
for (i = 4; i <= n; i += 2) {
        telek[i - 1][3] = telek[i - 3][3] + telek[i - 3][1];
}

telek[0][3] = 0; //páratlan házszámok (Gazdagsor)
for (i = 3; i <= n; i += 2) {
        telek[i - 1][3] = telek[i - 3][3] + telek[i - 3][1];
}

int joletHossz = telek[n - 1][3] + telek[n - 1][1];
int gazdagHossz = telek[n - 2][3] + telek[n - 2][1];

System.out.println((gazdagHossz + joletHossz + 2 * oldal) + " m.\n");

System.out.println("A 3. feladat megoldása");
System.out.print("A teljes utcafront beépítések száma Jólétsoron: ");
int uf = 0;

for (i = 2; i <= n;  i += 2) {
        if (telek[i - 1][1] <= 20) {
                uf++;
        }
}

System.out.println(uf);

System.out.println("\nA 4. feladat megoldása");
System.out.print("A legkisebb és a legnagyobb területű ház között Gazdagsoron ");
int min = 1;
int max = 1;
for (i = 1; i <= n; i += 2) {
        if (telek[i - 1][1] * telek[i - 1][2] < telek[min - 1][1] * telek[min - 1][2]) {
                min = i;
        }
        if (telek[i - 1][1] * telek[i - 1][2] > telek[max - 1][1] * telek[max - 1][2]) {
                max = i;
        }
}
System.out.println((Math.abs(max - min) - 1) + " ház helyezkedik el.");
System.out.print("A legkisebb területű telek házszáma: " + telek[min - 1][0]);
System.out.println(", területe: " + telek[min - 1][1] * telek[min - 1][2] + " nm");
System.out.print("A legnagyobb területű telek házszáma: " + telek[max - 1][0]);
System.out.println(", területe: " + telek[max - 1][1] * telek[max - 1][2] + " nm");

System.out.println("");

System.out.println("Az 5. feladat megoldása");
System.out.println("Telekadó");
int telekado = 0;
for (i = 1; i <= n; i += 2) {
        telekado += adokivetes(telek[i - 1][1], telek[i - 1][2]);
}
System.out.println("Gazdagsoron az önkormányzat " + telekado + " Fabatka telekadó bevételre számíthat. \n");

System.out.println("A 6. feladat megoldása ");
System.out.println("Az utolsó három ház Jólétsoron: ");

for (i = n; i >= n - 4; i -= 2) {
        System.out.print("Házszám: " + telek[i - 1][0]);
        System.out.println(", a telek távolsága Jólétsor elejétől: " + telek[i - 1][3] + " m");

}
System.out.println("");
```

```
                System.out.println("A 7. feladat megoldása");
                System.out.println("Jólétsor telkeinek hossza, kiíratás a joletsor.csv fájlba \n");

                PrintWriter kiir = new PrintWriter(new FileWriter("joletsor.csv"));

                int patak = 10;
                int minHossz;

                for (i = 2; i <= n; i += 2) { //Jólétsor
                        minHossz = oldal;

                        for (j = 1; j <= n; j +=2) {//Gazdagsor
                                if (szemben(telek[i - 1], telek[j - 1])) {
                                        // A tömböket adunk át a függvénynek
                                        if (oldal - patak - telek[j - 1][2] < minHossz) {
                                                // az irreálisan nagy kezdőértékre biztosan teljesül
                                                minHossz = oldal - patak - telek[j - 1][2];
                                        }
                                }
                        }
                        kiir.println(telek[i - 1][0] + ";" + telek[i - 1][1] + ";" + minHossz);
                }

                kiir.close();

                System.out.println("A fájlkiírás befejeződött. \n");
        }

        public static int adokivetes(int szel, int hossz) {
                int terulet = szel * hossz;
                int ado;

                if (terulet <= 700) {
                        ado = terulet * 51;
                } else if (terulet <= 1000) {
                        ado = 700 * 51 + (terulet - 700) * 39;
                } else {
                        ado = 700 * 51 + 300 * 39 + 200;
                }
                if (szel <= 15 || hossz <= 25) {
                        ado = (int) (ado * 0.8);
                }
                int apro = ado % 100;
                ado -= apro;
                if (apro >= 50) {
                        ado += 100;
                }
                return ado;
        }

        public static boolean szemben(int[] j, int[] g) {
                /*
                 * Az [a;b] és [c;d] intervallumoknak akkor van közös pontja,
                 * ha d >= a és c <= b
                 */
                return ((g[3] + g[1] >= j[3]) && (g[3] <= j[3] + j[1]));
        }

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása az telkek.txt fájlból
A beolvasás megtörtént.

A 2. feladat megoldása
Egy körséta hossza: 920 m.

A 3. feladat megoldása
A teljes utcafront beépítések száma Jólétsoron: 8

A 4. feladat megoldása
A legkisebb és a legnagyobb területű ház között Gazdagsoron 11 ház helyezkedik el.
A legkisebb területű telek házszáma: 15, területe: 600 nm
A legnagyobb területű telek házszáma: 27, területe: 1225 nm

Az 5. feladat megoldása
Telekadó
Gazdagsoron az önkormányzat 629100 Fabatka telekadó bevételre számíthat.

A 6. feladat megoldása
Az utolsó három ház Jólétsoron:
Házszám: 32, a telek távolsága Jólétsor elejétől: 360 m
Házszám: 30, a telek távolsága Jólétsor elejétől: 325 m
Házszám: 28, a telek távolsága Jólétsor elejétől: 300 m

A 7. feladat megoldása
Jólétsor telkeinek hossza, kiíratás a joletsor.csv fájlba

A fájlkiírás befejeződött.
```

A joletsor.csv szöveges fájl

```
2;15;45
4;30;25
6;25;15
8;20;25
10;15;30
12;30;30
14;35;35
16;20;40
18;15;45
20;30;25
22;25;20
24;20;25
26;20;35
28;25;35
30;35;30
32;20;40
```

2010 október: Anagramma

Az anagramma a szójátékok egy fajtája, melyben értelmes szavak vagy mondatok betűinek sorrendjét úgy változtatjuk meg, hogy az eredmény szintén értelmes szó vagy mondat lesz. Sok anagramma esetén az eredeti szó és a végeredmény között humoros vagy egyéb kapcsolat van, ez növeli az anagramma érdekességét, értékét. Például a satu szó anagrammái: utas, tusa, suta.

A *szotar.txt* ASCII kódolású állomány legfeljebb 300 különböző szót tartalmaz. A szavak legalább 2, legfeljebb 30 karakter hosszúságúak, és csak az angol ábécé kisbetűit tartalmazzák. Az állományban az egyes szavak külön sorokban szerepelnek, és minden szó csak egyszer fordulhat elő.

Például:

szotar.txt

```
eszesen
kereszt
keretes
keretez
nyertesek
hadartam
maradhat
…
```

Készítsen programot, amely az alábbi kérdésekre válaszol! A program forráskódját *anagram* néven mentse! Ügyeljen arra, hogy programjának minden helyes tartalmú bemeneti állomány esetén működnie kell!

Minden részfeladat megoldása előtt írja a képernyőre a feladat sorszámát! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár (például az 1. feladat esetén: „`Adja meg a szöveget:`”)! A képernyőn megjelenített üzenetek esetén az ékezetmentes kiírás is elfogadott.

1. Kérjen be a felhasználótól egy szöveget, majd határozza meg, hogy hány különböző karakter található a szövegben! A darabszámot és a karaktereket írja ki a képernyőre!
2. Olvassa be a *szotar.txt* állományból a szavakat, és a következő feladatok megoldása során ezekkel dolgozzon! Amennyiben nem tudja beolvasni az állományból a szavakat, akkor az első 10 szóval dolgozzon!
3. Az állományból beolvasott szavakat alakítsa át úgy, hogy minden szó karaktereit egyenként tegye ábécérendbe! Az így létrehozott szavakat írja ki az *abc.txt* állományba az eredeti állománnyal egyező sorrendben!

   Például:

Eredeti	Ábécé sorrendben lévő
tervez	eertvz
nyugalom	aglmnouy

4. Kérjen be a felhasználótól két szót, és döntse el, hogy a két szó anagramma-e! Ha azok voltak, írja ki a képernyőre az „`Anagramma`” szót, ha nem, akkor pedig a „`Nem anagramma`” szöveget!
5. Kérjen be a felhasználótól egy szót! A *szotar.txt* állomány szavaiból keresse meg a szó anagrammáit (a szót önmagát is annak tekintve)! Ha van találat, azokat egymás alá írja ki a képernyőre, ha nem volt találat, akkor írja ki a „`Nincs a szótárban anagramma`” szöveget!

6. Határozza meg, hogy a *szotar.txt* állományban melyik a leghosszabb szó! Ha több, ugyanannyi karakterből álló leghosszabb szó volt, akkor az ugyanazokat a karaktereket tartalmazó szavakat (amelyek egymás anagrammái) közvetlenül egymás alá írja ki! A feltételnek megfelelő összes szó pontosan egyszer szerepeljen a kiírásban!

7. Rendezze a *szotar.txt* állományban lévő szavakat a karakterek száma szerint növekvő sorrendbe! Az egyforma hosszúságú és ugyanazokat a karaktereket tartalmazó szavak (amelyek egymás anagrammái) szóközzel elválasztva ugyanabba a sorba kerüljenek! Az egyforma hosszúságú, de nem ugyanazokat a karaktereket tartalmazó szavak külön sorba kerüljenek! A különböző hosszúságú szavakat egy üres sorral különítse el egymástól! Az így rendezett szavakat írja ki a *rendezve.txt* állományba!

Például:

Eredeti	Rendezett
halat rakat ajak papi rakta ajka takar kaja satu vallat tarka pipa paplan	ajak ajka kaja papi pipa satu suta tusa utas halat rakat rakta takar tarka vallat paplan

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Anagramma
 *
 * @author Klemand
 */

public class EmeltInfo2010okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");

		System.out.print("Kérek egy szót! ");

		Scanner sc = new Scanner(System.in);
		String szo = sc.nextLine();

		int i, j;
		String betuHalmaz = "";
		String betu;
		for (i = 1; i <= szo.length(); i++) {
			betu = szo.substring(i - 1, i);
			if (!(betuHalmaz.contains(betu))) {
				betuHalmaz += betu;
			}
		}

		System.out.print("\nA beírt szó " + betuHalmaz.length() + " különböző karaktert tartalmaz: ");
		for (i = 1; i <= betuHalmaz.length(); i++) {
			System.out.print(betuHalmaz.substring(i - 1, i) + " ");
		}
		System.out.println("\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("A szotar.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("szotar.txt"));

		String[] szavak = new String[300];
		String sor;
		int db = 0;
		while ((sor = beolvas.readLine()) != null) {
			db++;
			szavak[db - 1] = sor;
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");
```

```
System.out.println("A 3. feladat megoldása");

System.out.println("A  karakterek szerint rendezett szavak kiíratása az abc.txt fájlba");

String[] rendSzavak = new String[300];
for (i = 1; i <= db; i++) {
        rendSzavak[i - 1] = karRendezes(szavak[i - 1]);
}

PrintWriter kiir = new PrintWriter(new FileWriter("abc.txt"));

for (i = 1; i <= db; i++) {
        kiir.println(rendSzavak[i - 1]);
}
kiir.close();
System.out.println("A fájlkiírás befejeződött. \n");

System.out.println("A 4. feladat megoldása");
String szo1, szo2;
System.out.println("Két bekért szó összehasonlítása");
System.out.print("Kérem az első szót: ");
szo1 = sc.nextLine();
System.out.print("Kérem a második szót: ");
szo2 = sc.nextLine();

if (karRendezes(szo1).equals(karRendezes(szo2))) {
        System.out.println("Anagramma");
} else {
        System.out.println("Nem anagramma");
}
System.out.println("");

System.out.println("Az 5. feladat megoldása");

System.out.println("Egy bekért szó anagrammái a szótárban");
System.out.print("Kérem a szót: ");
szo = sc.nextLine();
String rendSzo = karRendezes(szo);
boolean van = false;

for (i = 1; i <= db; i++) {
        if (rendSzavak[i - 1].equals(rendSzo)) {
                if (!van) {
                        System.out.println("A szó anagrammái a szótárban:");
                        // Ezt csak az első találatnál írja ki.
                        van = true;
                }
                System.out.println(szavak[i - 1]);
        }
}
if (!van) {
        System.out.println("Nincs a szótárban anagramma");
}
```

```
System.out.println("\nA 6. feladat megoldása");

System.out.println("A leghosszabb szavak anagrammáik szerint csoportosítva:");

// Lista készítése a rendezett szavakról
String[] rendSzoLista = new String[300];
int listaDb = 0;
String aktRendSzo;
for (i = 1; i <= db; i++) {
        aktRendSzo = rendSzavak[i - 1];
        j = 1;
        while ((j <= listaDb) && !(aktRendSzo.equals(rendSzoLista[j - 1]))) {
                j++;
        }

        if (j > listaDb) {
                listaDb++;
                rendSzoLista[listaDb - 1] = aktRendSzo;
        }
}

// A lista rendezése a szavak hossza szerint
String asztal;

for (i = listaDb - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (rendSzoLista[j - 1].length() > rendSzoLista[j].length()) {
                        asztal = rendSzoLista[j - 1];
                        rendSzoLista[j - 1] = rendSzoLista[j];
                        rendSzoLista[j] = asztal;
                }
        }
}

int n = listaDb;
// Végigmegyünk (visszafelé) a rendezett leghosszabb szavakon
int maxHossz = rendSzoLista[n - 1].length();
i = n;
while (rendSzoLista[i - 1].length() == maxHossz) {
        // és mindegyikhez kiíratjuk a szótárban szereplő anagrammáit
        aktRendSzo = rendSzoLista[i - 1];
        for (j = 1; j <= db; j++) {
                if (rendSzavak[j - 1].equals(aktRendSzo)) {
                        System.out.println(szavak[j - 1]);
                }
        }
        System.out.println("");
        i--;
}

sc.close();
```

```
                System.out.println("A 7. feladat megoldása");

                System.out.println("A szavak hossza szerint rendezett szótár kiíratása a rendezve.txt fájlba");
                // Az előbb csak a leghosszabb szavak kellettek a listából, most az összes.

                kiir = new PrintWriter(new FileWriter("rendezve.txt"));

                for (i = 1; i <= listaDb; i++) {
                        for (j = 1; j <= db; j++) {
                                aktRendSzo = rendSzoLista[i - 1];
                                for (j = 1; j <= db; j++) {
                                        if (rendSzavak[j - 1].equals(aktRendSzo)) {
                                                kiir.print(szavak[j - 1] + " ");
                                        }
                                }
                        }
                        kiir.println();
                        if (i < listaDb && (rendSzoLista[i - 1].length() < rendSzoLista[i].length())) {
                                kiir.println();
                        }
                }

                kiir.close();
                System.out.println("A fájlkiírás befejeződött. \n");

        }

        public static String karRendezes(String szo) {
                // buborékos rendezés

                int hossz = szo.length();
                int i, j;
                String asztal;
                String[] daraboltSzo = szo.split("");

                for (i = hossz - 1; i >= 1; i--) {
                        for (j = 1; j <= i; j++) {
                                if (daraboltSzo[j - 1].compareTo(daraboltSzo[j]) > 0) {
                                        asztal = daraboltSzo[j - 1];
                                        daraboltSzo[j - 1] = daraboltSzo[j];
                                        daraboltSzo[j] = asztal;
                                }
                        }
                }

                String rendezett = "";
                for (i = 1; i <= hossz; i++) {
                        rendezett += daraboltSzo[i - 1];
                }
                return rendezett;
        }

}
```

```
Az 1. feladat megoldása
Kérek egy szót! naplemente

A beírt szó 7 különböző karaktert tartalmaz: n a p l e m t

A 2. feladat megoldása
A szotar.txt fájl beolvasása
A beolvasás megtörtént.

A 3. feladat megoldása
A  karakterek szerint rendezett szavak kiíratása az abc.txt fájlba
A fájlkiírás befejeződött.

A 4. feladat megoldása
Két bekért szó összehasonlítása
Kérem az első szót: nyugat
Kérem a második szót: nyugtat
Nem anagramma

Az 5. feladat megoldása
Egy bekért szó anagrammái a szótárban
Kérem a szót: alvokat
A szó anagrammái a szótárban:
alkotva
lovakat
vakolat

A 6. feladat megoldása
A leghosszabb szavak anagrammáik szerint csoportosítva:
kalandtura

hajnalokat
hatoljanak

A 7. feladat megoldása
A szavak hossza szerint rendezett szótár kiíratása a rendezve.txt fájlba
A fájlkiírás befejeződött.
```

A szotar.txt forrásfájl

```
eszesen
kereszt
keretes
keretez
nyertesek
hadartam
maradhat
hajnalokat
bajsza
szabja
ajkaid
kiadja
dalnak
kaland
lankad
alighanem
meghalnia
fogalmam
fogammal
faragott
forgatta
falait
fiatal
...
```

Az abc.txt szövegfájl

```
eeenssz
eekrstz
eeekrst
eeekrtz
eeeknrsty
aaadhmrt
aaadhmrt
aaahjklnot
aabjsz
aabjsz
aadijk
aadijk
aadkln
aadkln
aadkln
aaeghilmn
aaeghilmn
aafglmmo
aafglmmo
aafgortt
aafgortt
aafilt
aafilt
...
```

A rendezve.txt szövegfájl

```
ajak ajka kaja
papi pipa
satu suta tusa utas
kuka akku

rakat rakta takar tarka
durva udvar
botor robot
korok orkok
pakol lapok polka
eltol letol elolt
forma amorf farom

bajsza szabja
ajkaid kiadja
dalnak kaland lankad
falait fiatal
nappal paplan
alanyt nyalta
borzas szobra
szagol szolga
alszik szikla
parton pontra torpan
ketyeg tegyek
istene sietne
elkelt kellet lelket leltek

eszesen
kereszt
keretes
keretez
alhatom hatalom
szavait tavaszi
akarunk uraknak
alkotva lovakat vakolat
karomat takarom
hordtak korhadt
hasznos hosszan
korokat orkokat
didereg eddigre
fedelet feledte lefedte
karomba abrakom

hadartam maradhat
fogalmam fogammal
faragott forgatta
hangokra harangok
alkothat halottak holtakat
napokkal pakolnak
alattunk tanultak
boldogan dologban
koromban romokban
kapdosni kispadon
alkoholt hallotok

nyertesek
alighanem meghalnia
csontokra roncsokat
miniszter miszerint

hajnalokat hatoljanak
kalandtura
```

2011 május: Szójáték

Sok szórakoztató szójátékkal lehet elütni az időt. Ezek közül némelyekhez segítségül hívhatjuk a technikát is. Az alábbiakban szójátékokhoz kapcsolódó problémákat kell megoldania.

A feladatok megoldásához rendelkezésére áll a *szoveg.txt* fájl, amelybe Gárdonyi Géza Egri csillagok című regényéből gyűjtöttünk ki szavakat. Az állományban csak olyan szavak szerepelnek, melyek az angol ábécé betűivel leírhatók, és minden szó csak egyszer szerepel. A könnyebb feldolgozhatóság érdekében valamennyi szó csupa kisbetűvel szerepel, szavanként külön sorban. Tudjuk, hogy ebben az állományban a szavak 20 karakternél nem hosszabbak.

Készítsen programot, amely az alábbi feladatokat megoldja! A program forráskódját *szavak* néven mentse!

Minden – képernyőre írást igénylő – részfeladat megoldása előtt írja a képernyőre a feladat sorszámát! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár (például a 1. feladat esetén: „`1. feladat Adjon meg egy szót: `”)! Az ékezetmentes kiírás is elfogadott.

1. Kérjen be a felhasználótól egy szót, és döntse el, hogy tartalmaz-e magánhangzót!
   Amennyiben tartalmaz, írja ki, hogy „`Van benne magánhangzó.`”!
   Ha nincs, akkor írja ki, hogy „`Nincs benne magánhangzó.`”! A begépelendő szóról feltételezheti, hogy csak az angol ábécé kisbetűit tartalmazza.
   (Az angol ábécé magánhangzói: a, e, i, o, u.)
2. Írja ki a képernyőre, hogy melyik a leghosszabb szó a *szoveg.txt* állományban, és az hány karakterből áll! Ha több azonos leghosszabb hosszúságú szó is van a szógyűjteményben, akkor azok közül elegendő egyetlen szót kiírnia. A feladatot úgy oldja meg, hogy tetszőleges hosszúságú szövegállomány esetén működjön, azaz a teljes szöveget ne tárolja a memóriában!
3. A magyar nyelv szavaiban általában kevesebb a magánhangzó, mint a mássalhangzó. Határozza meg, hogy az állomány mely szavaiban van több magánhangzó, mint egyéb karakter! Ezeket a szavakat írja ki a képernyőre egy-egy szóközzel elválasztva! A szavak felsorolása után a mintának megfelelően az alábbi adatokat adja meg:

   - hány szót talált;
   - hány szó van összesen az állományban;
   - a talált szavak hány százalékát teszik ki az összes szónak!

   A százalékot két tizedessel szerepeltesse!

   Például:

   ```
   130/3000 : 4,33%
   ```

A következőkben a szólétra játékkal kapcsolatos feladatokat kell megoldania.

A szólétra építés egy olyan játék, amikor adott egy szó közepe, például *isz*, amit a létra fokának nevezünk. Ennek a szócsonknak az elejére és a végére kell egy-egy betűt illesztenünk úgy, hogy értelmes szót hozzunk létre, például *hiszi* vagy *liszt*. Ezt az értelmes szót a játékban létraszónak nevezzük. Az adott szórészlethez minél több létraszót tudunk kitalálni, annál magasabb lesz a szólétra. A cél az, hogy egy megadott szócsonkhoz a lehető legmagasabb szólétrát építsük.

Például:

Szórészlet: **isz**

A hozzá tartozó létraszavak:

h**isz**i
l**isz**t
v**isz**i
t**isz**t
…

4. Hozzon létre egy tömb vagy lista adatszerkezetet, és ebbe gyűjtse ki a fájlban található ötkarakteres szavakat! A *szoveg.txt* állomány legfeljebb 1000 darab ötkarakteres szót tartalmaz. Kérjen be a felhasználótól egy 3 karakteres szórészletet! Írja ki a képernyőre a szólétra építés szabályai szerint hozzá tartozó ötkarakteres szavakat a tárolt adathalmazból! A kiírásnál a szavakat egy-egy szóköz válassza el! (Teszteléshez használhatja például az „isz” vagy „obo” szórészleteket, mert ezekhez a megadott szövegállományban több létraszó is tartozik.)

5. Az eltárolt ötkarakteres szavakból csoportosítsa azokat a szavakat, melyek ugyanannak a hárombetűs szórészletnek a létraszavai! Hozzon létre egy *letra.txt* állományt, amelybe ezeket a szavakat írja az alábbiak szerint:

   - minden szó külön sorba kerüljön;
   - csak olyan szó szerepeljen az állományban, aminek van legalább egy párja, amivel egy létrát alkotnak (azaz első és utolsó karakter nélkül megegyeznek); • az egy létrához tartozó szavak közvetlenül egymás után helyezkedjenek el;
   - két létra szavai között egy üres elválasztó sor legyen!

Például:

letra.txt

```
megye
vegye
hegyi
tegye

lehet
teher
mehet

tejes
fejet
fejen

neked
nekem
reked

...
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Szójáték
 *
 * @author Klemand
 */
public class EmeltInfo2011maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása ");
		System.out.println("Van-e magánhangzó egy bekért szóban?");

		System.out.print("Kérek egy szót: ");

		Scanner sc = new Scanner(System.in);
		String fok = sc.nextLine().toLowerCase();
		int mghDb = mghDbMeghat(fok);

		if (mghDb > 0) {
			System.out.println("Van benne magánhangzó.");
		} else {
			System.out.println("Nincs benne magánhangzó.");
		}

		System.out.println("\nA 2. feladat megoldása");
		System.out.println("A szoveg.txt fájl leghosszabb szavának meghatározása");
		System.out.println("a szavak belső tárolása nélkül");

		BufferedReader beolvas = new BufferedReader(new FileReader("szoveg.txt"));

		String sor = beolvas.readLine();
		int db = 1;
		String leghosszabb = sor;
		int maxhossz = sor.length();

		while ((sor = beolvas.readLine()) != null) {
			db++;
			if (sor.length() > maxhossz) {
				leghosszabb = sor;
				maxhossz = sor.length();
			}
		}
		beolvas.close();

		System.out.println("A leghosszabb szó: " + leghosszabb + ", hossza: " + maxhossz + " karakter.");
```

```
System.out.println("\nA 3. feladat megoldása");

System.out.println("A szoveg.txt fájl magánhangzó túlsúlyú szavai: ");

beolvas = new BufferedReader(new FileReader("szoveg.txt"));
int tobbMghDb = 0;

int i;
for (i = 1; i <= db; i++) {
        sor = beolvas.readLine();
        if (sor.length() < 2 * mghDbMeghat(sor)) {
                System.out.print(sor + " ");
                tobbMghDb++;
                if (tobbMghDb % 10 == 0) {
                        System.out.println("");
                }
        }
}
System.out.println();
beolvas.close();

System.out.print(tobbMghDb + "/" + db + " : ");
double szazalek = ((double) tobbMghDb / db) * 100;
System.out.printf("%.2f", szazalek);
System.out.println("% \n");

System.out.println("A 4. feladat megoldása");
System.out.println("Az ötkarakteres szavak beolvasása a szoveg.txt fájlból.");

beolvas = new BufferedReader(new FileReader("szoveg.txt"));

String[] szavak5 = new String[1000];
int db5 = 0;

for (i = 1; i <= db; i++) {
        sor = beolvas.readLine();
        if (sor.length() == 5) {
                db5++;
                szavak5[db5 - 1] = sor;
        }
}
beolvas.close();
System.out.println("A beolvasás megtörtént.");
System.out.println("A fájlban összesen " + db5 + " ötkarakteres szó található.");

System.out.println("Egy bekért szó létraszavai a beolvasott ötkarakteres szavak között");
System.out.print("Kérek egy 3 karakteres ékezetmentes szót: ");
fok = sc.nextLine();
String aktFok;
for (i = 1; i <= db5; i++) {
        aktFok = szavak5[i - 1].substring(1, 4);
        if (aktFok.equals(fok)) {
                System.out.print(szavak5[i - 1] + " ");
        }
}
System.out.println("\n");
sc.close();
```

```
                System.out.println("Az 5. feladat megoldása");

                System.out.println("A létraszavak kiíratása csoportosítva a letra.txt fájlba:");

                // A létrafokok gyakorisági táblázatának elkészítése
                String[] fokLista = new String[1000];
                int[] gyakorisag = new int[1000];
                int listaDb = 0;
                int j;
                for (i = 1; i <= db5; i++) {
                        aktFok = szavak5[i - 1].substring(1, 4);
                        j = 1;
                        while ((j <= listaDb) && !(aktFok.equals(fokLista[j - 1]))) {
                                j++;
                        }

                        if (j <= listaDb) {
                                gyakorisag[j - 1]++;
                        } else {
                                listaDb++;
                                fokLista[listaDb - 1] = aktFok;
                                gyakorisag[listaDb - 1] = 1;
                        }
                }

                PrintWriter kiir = new PrintWriter(new FileWriter("letra.txt"));

                for (i = 1; i <= listaDb; i++) {
                        if (gyakorisag[i - 1] > 1) {
                                for (j = 1; j <= db5; j++) {
                                        aktFok = szavak5[j - 1].substring(1, 4);
                                        if (aktFok.equals(fokLista[i - 1])) {
                                                kiir.println(szavak5[j - 1]);
                                        }
                                }
                                if (i < listaDb) {
                                        kiir.println();
                                }
                        }
                }
                kiir.close();
                System.out.println("Az letra.txt fájl kiíratása befejeződött. \n");

        }

        public static int mghDbMeghat(String szo) {

                String mgh = "aeiou";
                int hossz = szo.length();
                int db = 0;

                for (int i = 1; i <= hossz; i++) {
                        if (mgh.contains(szo.substring(i - 1, i))) {
                                db++;
                        }
                }
                return db;
        }

}
```

```
Az 1. feladat megoldása
Van-e magánhangzó egy bekért szóban?
Kérek egy szót: naplemente
Van benne magánhangzó.

A 2. feladat megoldása
A szoveg.txt fájl leghosszabb szavának meghatározása
a szavak belső tárolása nélkül
A leghosszabb szó: angyalszobrokat, hossza: 15 karakter.

A 3. feladat megoldása
A szoveg.txt fájl magánhangzó túlsúlyú szavai:
ablakai aga ahova akarata aki ali alias amade ami amije
amire amoda anyai apa atyai bibliai budai dalia dunai eledele
eleinte eleje emberei emeleti emeletieket emeli ennie erdei ereiben ereje
ezrei falai fia fiai fiaihoz fiaim fiaimat fiaira fiait fogai
fokai haramia ide ideadta idegein idei ideig ideje idomait igazi
ilona innia katonai kezei kiemeli korai levelei lovai mai mezei
oda odaadja odaadok odaadom odaadta odahaza odasiet odaveti oka ormai
orvosai piaca piaci rokonai sugaraiba ujjai ura uraim utcai
79/7825 : 1,01%

A 4. feladat megoldása
Az ötkarakteres szavak beolvasása a szoveg.txt fájlból.
A beolvasás megtörtént.
A fájlban összesen 756 ötkarakteres szó található.
Egy bekért szó létraszavai a beolvasott ötkarakteres szavak között
Kérek egy 3 karakteres ékezetmentes szót: egy
hegye hegyi megye tegye vegye

Az 5. feladat megoldása
A létraszavak kiíratása csoportosítva a letra.txt fájlba:
Az letra.txt fájl kiíratása befejeződött.
```

A letra.txt szövegfájl

```
addig                   arcok                   bakta                   bokor
eddig                   arcon                   rakta                   fokot
                        arcot                                           pokol
adtak                                           balra
adtam                   arcuk                   falra                   borok
                        arcul                                           boros
agyag                                           barom                   forog
ugyan                   babot                   karok                   korom
                        habos                   karom                   koron
agyba                   rabok                   karon                   poros
egybe                   rabom                   marok                   sorom
                        rabon                   maros                   soron
akkor                   rabot                                           torok
ekkor                   zabot                   batyu
                                                matyi                   borul
aludj                   bajba                                           poruk
aludt                   hajba                   benne
                        lajbi                   lenne                   budai
amely                                           lenni                   cudar
emeli                   bajod                   menni                   ludak
emelt                   bajom                   tenni                   rudat
                        bajos                   venni
annak                   hajol                                           cecey
onnan                   lajos                   beteg                   pecek
                        vajon                   hetet
annyi                   zajos                   vetem                   cicus
ennyi                                                                   vicus
                                                                        ...
```

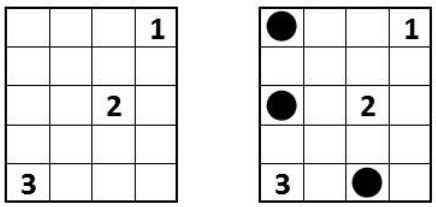
2011 május idegennyelvű: Rejtvény

Egy weboldalon érdekes rejtvényt tesznek közzé hétről hétre. A rejtvényekben egy N×Mes területre világítótornyokat helyeznek le. Ezeket a tornyokat számmal jelölik. Minden alkalommal az a feladat, hogy a területre el kell helyezni X darab hajót úgy, hogy minden toronyból (vízszintesen és függőlegesen összesen) annyi hajó legyen látható, ahányas szám a tornyot jelképező mezőben van!

A hajókra vonatkozó szabályok a következők:

- Minden hajó egy négyzet nagyságú.
- A hajók nem érintkezhetnek egymással, még átlós irányban sem.
- A hajók nem érinthetik a világítótornyokat, még átlós irányban sem.
- A hajók egymást nem takarják ki. Azaz a világítótoronyból az egy vonalban lévő hajók is látszanak.

Például:
Egy 5×4-es terület és 3 hajó esetén



A weboldalon ugyanúgy, mint az előző hetekben, egy 10×10-es négyzetbe kell elhelyezni 12 darab hajót. A versenyzők által beküldött megfejtéseket alkalmazás segítségével összefűzik egy txt állományba. Ennek a fájlnak az első sora a megfejtések számát tartalmazza, ami maximálisan 20 darab lehet. Minden megfejtés előtt pedig a megfejtő neve található. Az egyes megfejtésekben a vizet 0-val, a világítótornyot egy 1 és 9 közötti számmal, a hajókat pedig 11-es számmal jelölik. A fájlban a számokat egy-egy szóközzel választják el.

Például:

A *megoldas.txt* állomány egy részlete. (A példát szabályos táblázatban jelenítjük meg a jobb átláthatóság érdekében.)

```
10
Absolon
 0  0  0  0 11  0 11  0  0  0
11  0  2  0  0  0  0  0  0 11
 0  0  0  0  0  1  0 11  0  0
 0  0  0 11  0  0  0  0  0  0
 0  3  0  0  0  0  0  0 11  0
 0  0  0  0  2  0 11  0  0  0
 0 11  0  0  0  0  0  0  3  0
 0  0  0  0  0  0  3  0  0  0
 0 11  0  3  0  0  0  0 11  0
 0  0  0  0  0  0 11  0  0  0
…
```

A 2. sor 3. oszlopában tehát egy világítótorony van, amelynek sorában és oszlopában összesen 2 hajó lehet. A 2. sor 1. oszlopában és a 2. sor 10. oszlopában egy-egy hajó található.

Készítsen programot, amely a rejtvényre érkező megoldások helyességét ellenőrzi!
A program forráskódját *rejtveny* néven mentse!

Minden feladat megoldása előtt írja a képernyőre a feladat sorszámát! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár (például az 1. feladat esetén: „`Adja meg a torony adatait!`”)! Az ékezetmentes kiírás is elfogadott.

A feladatok megoldása során feltételezzük, hogy a beolvasott adatok helyesek, ezért azokat sehol nem kell ellenőrizni!

1. Kérje be a felhasználótól egy 10×10-es táblára vonatkozóan egy világítótorony pozícióját (a torony helyének sor és oszlop száma), és a toronyból látható hajók számát! A rejtvény megfejtését a nagy számmal rendelkező tornyoknál érdemes kezdeni. Ezért, ha a torony értéke nagyobb, mint három, akkor írja ki a képernyőre, hogy „`Nehéz torony.`”, más esetben ne írjon ki semmit!
2. A megadott világítótorony helyzete alapján állapítsa meg, hogy a szabályok szerint a világítótorony körül mely helyekre biztosan nem kerülhet hajó! Az eredményt írassa ki a képernyőre úgy, hogy a tiltott helyek sor és oszlop azonosítói vesszővel elválasztva külön sorokba kerüljenek! Például ha a világítótorony a 2, 3 pozícióban van, akkor:

```
1,2
1,3
1,4
2,2
2,4
3,2
3,3
3,4
```

3. A *feladvany.txt* állomány tartalmazza az erre a hétre kiadott rejtvényt a már ismert formában. Olvassa be a rejtvényt az állományból és a *megoldas.txt* állományban beküldött megoldások közül szűrje ki azokat, amelyek nem az e heti feladványra érkeztek. Ezen megfejtő(k) nevét írja ki a képernyőre! Ha minden megfejtés az e heti feladványra érkezett, akkor írja ki a képernyőre, hogy „`Mindegyik megoldás erre a heti feladványra érkezett.`”!
4. Azok közül a megoldások közül, amelyek erre a heti feladványra érkeztek, állapítsa meg, hogy melyikekben van kevesebb vagy több hajó megadva, mint 12! Írja ki a képernyőre, hogy e szempontból hány darab hibás „megoldás” volt!
5. Hány olyan szabálytalan megoldás született az e heti feladatra, amelyben:
   - a hajók száma megfelelő és
   - egy vagy több hajó elhelyezése a szomszédsági kapcsolatokra vonatkozó szabályoknak nem megfelelő?

   Az eredményt írja ki a képernyőre!
6. Határozza meg, hogy hány megoldás volt helyes a beküldöttek közül! Az ellenőrzésnél vegye figyelembe az előző pontokban leírtakat, valamint azt, hogy a világítótornyok az értéküknek megfelelő számú hajót látnak-e! A helyes beküldők nevét írja ki a képernyőre!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

/**
 * Rejtvény
 *
 * @author Klemand
 */
public class EmeltInfo2011mid {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Egy beolvasott torony nehézsége");

		Scanner sc = new Scanner(System.in);

		System.out.println("Kérem a torony adatait!");
		System.out.print("A sor száma (max. 10): ");
		int sor = sc.nextInt();
		System.out.print("Az oszlop száma (max. 10): ");
		int oszlop = sc.nextInt();
		System.out.print("A látható hajók száma (max. 9): ");
		int lathatoHajo = sc.nextInt();
		if (lathatoHajo > 3) {
			System.out.println("Nehéz torony.");
		}
		sc.close();

		System.out.println("\nA 2. feladat megoldása");
		System.out.println("A hajók számára tiltott mezők a torony körül:");

		int i, j;
		for (i = sor - 1; i <= sor + 1; i++) {
			for (j = oszlop - 1; j <= oszlop + 1; j++) {
				// a torony környezete
				if (i >= 1 && i <= 10 && j >= 1 && j <= 10 && !(i == sor && j == oszlop)) {
					// a tiltott hely a táblán legyen, és ne maga a torony
					System.out.println(i + "," + j);
				}
			}
		}

		System.out.println("\nA 3. feladat megoldása");
		System.out.println("A feladvany.txt és a megoldasok.txt fájlok beolvasása ");

		BufferedReader beolvas = new BufferedReader(new FileReader("feladvany.txt"));

		String fsor;
		String[] daraboltSor;
		int[][] feladvany = new int[10][10];

		for (i = 1; i <= 10; i++) {
			daraboltSor = beolvas.readLine().split(" ");
			for (j = 1; j <= 10; j++) {
				feladvany[i - 1][j - 1] = Integer.parseInt(daraboltSor[j - 1]);
			}
		}
		beolvas.close();
```

```
beolvas = new BufferedReader(new FileReader("megoldas.txt"));

int db, n;
fsor = beolvas.readLine();
db = Integer.parseInt(fsor); // a megoldások száma

int[][][] megoldas = new int[db][10][10];
// megoldas[sorszám: n][sor: i][oszlop: j]
String[] nev = new String[db];

for (n = 1; n <= db; n++) {
        fsor = beolvas.readLine();
        nev[n - 1] = fsor;

        for (i = 1; i <= 10; i++) {
                daraboltSor = beolvas.readLine().split(" ");
                for (j = 1; j <= 10; j++) {
                        megoldas[n - 1][i - 1][j - 1] = Integer.parseInt(daraboltSor[j - 1]);
                }
        }
}

beolvas.close();
System.out.println("A beolvasás megtörtént. A beküldött megoldások száma: " + db);

System.out.println("A nem e heti feladványra érkező megoldások kiszűrése");

boolean[] kiesett = new boolean[db];
for (n = 1; n <= db; n++) {
        kiesett[n - 1] = false;
}

int nemEHetiDb = 0;
boolean azonos;
for (n = 1; n <= db; n++) {
        azonos = true;
        i = 1;
        while (i <= 10 && azonos) {
                j = 1;
                while (j <= 10 && ((megoldas[n - 1][i - 1][j - 1] % 11) == feladvany[i - 1][j - 1])) {
                        // A 11-es maradék a hajókat lenullázza, a feladványt adja vissza
                        j++;
                }
                if (j <= 10) {
                        azonos = false;
                }
                i++;
        }
        if (!azonos) {
                nemEHetiDb++;
                System.out.println(nev[n - 1]);
                kiesett[n - 1] = true;
        }
}

if (nemEHetiDb == 0) {
        System.out.println("Mindegyik megoldás erre a heti feladványra érkezett.");
}else {
        System.out.println("Számuk: " + nemEHetiDb);
}
```

```
System.out.println("\nA 4. feladat megoldása");
System.out.println("A 12-nél kevesebb vagy több hajót beküldők a heti megfejtők között:");

int hajoDb;
int hibasDb = 0;

for (n = 1; n <= db; n++) {
        hajoDb = 0;
        if (!kiesett[n - 1]) {
                for (i = 1; i <= 10; i++) {
                        for (j = 1; j <= 10; j++) {
                                if (megoldas[n - 1][i - 1][j - 1] == 11) {
                                        hajoDb++;
                                }
                        }
                }
                if (!(hajoDb == 12)) {
                        hibasDb++;
                        System.out.println(nev[n - 1]);
                        kiesett[n - 1] = true;
                }
        }
}
System.out.println("Számuk: " + hibasDb);

System.out.println("\nAz 5. feladat megoldása");
System.out.println("Tiltott helyen álló hajót beküldők a még versenyben lévők között: ");

boolean vanTiltott;
hibasDb = 0;

for (n = 1; n <= db; n++) {
        if (!kiesett[n - 1]) {
                vanTiltott = false;
                i = 1;
                while (i <= 10 && !vanTiltott) {
                        j = 1;
                        while (j <= 10 && !vanETiltott(i, j, megoldas[n - 1])) {
                                j++;
                        }
                        if (j <= 10) {
                                vanTiltott = true;
                        }
                        i++;
                }
                if (vanTiltott) {
                        hibasDb++;
                        System.out.println(nev[n - 1]);
                        kiesett[n - 1] = true;
                }
        }
}
System.out.println("Számuk: " + hibasDb);
```

```
                System.out.println("\nA 6. feladat megoldása");
                System.out.println("A helyes megoldást beküldők a heti feladványra:");

                boolean annyitLat;
                int joDb = 0;

                for (n = 1; n <= db; n++) {
                        if (!kiesett[n - 1]) {
                                annyitLat = true;
                                i = 1;
                                while (i <= 10 && annyitLat) {
                                        j = 1;
                                        while (j <= 10 && annyitLatE(i, j, megoldas[n - 1])) {
                                                j++;
                                        }
                                        if (j <= 10) {
                                                annyitLat = false;
                                        }
                                        i++;
                                }
                                if(annyitLat) {
                                        joDb++;
                                        System.out.println(nev[n - 1]);
                                }
                        }
                }
                System.out.println("A helyes megfejtést beküldők száma: " + joDb +"\n");

        }

        public static boolean vanETiltott(int sor, int oszlop, int[][] megold) {

                boolean van = false;
                int u, v;
                int aktObj = megold[sor - 1][oszlop - 1];

                if ((aktObj) == 0) {
                        return false;
                } else {
                        for (u = sor - 1; u <= sor + 1; u++) {
                                for (v = oszlop - 1; v <= oszlop + 1; v++) {
                                        if (u >= 1 && u <= 10 && v >= 1 && v <= 10 && !(u == sor && v == oszlop))
{
                                                if (!(megold[u - 1][v - 1] * aktObj == 0)) {
                                                        van = true;
                                                }
                                        }
                                }
                        }
                        return van;
                }
        }

        public static boolean annyitLatE(int sor, int oszlop, int[][] megold) {

                int lathato = 0;
                int u, v;

                if ((megold[sor - 1][oszlop - 1]) % 11 != 0) {
                        // ha világítótorony
                        for (u = 1; u <= 10; u++) {
                                for (v = 1; v <= 10; v++) {
                                        if ((u == sor || v == oszlop) && (megold[u - 1][v - 1] == 11)) {
                                                lathato++;
                                        }
                                }
                        }
                        return (lathato == megold[sor - 1][oszlop - 1]);
                } else {
                        return true;
                }
        }

}
```

```
Az 1. feladat megoldása
Egy beolvasott torony nehézsége
Kérem a torony adatait!
A sor száma (max. 10): 9
Az oszlop száma (max. 10): 10
A látható hajók száma (max. 9): 4
Nehéz torony.

A 2. feladat megoldása
A hajók számára tiltott mezők a torony körül:
8,9
8,10
9,9
10,9
10,10

A 3. feladat megoldása
A feladvany.txt és a megoldasok.txt fájlok beolvasása
A beolvasás megtörtént. A beküldött megoldások száma: 10
A nem e heti feladványra érkező megoldások kiszűrése
Ildefonz
Ozor
Számuk: 2

A 4. feladat megoldása
A 12-nél kevesebb vagy több hajót beküldők a heti megfejtők között:
Absolon
Verner
Számuk: 2

Az 5. feladat megoldása
Tiltott helyen álló hajót beküldők a még versenyben lévők között:
Selton
Meliton
Számuk: 2

A 6. feladat megoldása
A helyes megoldást beküldők a heti feladványra:
Kasztor
Folkus
Bazil
A helyes megfejtést beküldők száma: 3
```

2011 október: Pitypang

A kerekdombi Pitypang wellness hotel nem régen nyitotta meg kapuit. A szállodában összesen 27 szoba van. A szobák egységesen kétágyasak, de minden szobában egy pótágy elhelyezésére is van lehetőség. Árképzés szempontjából három különböző időszakot határolt el a szálloda vezetősége: tavaszi, nyári és őszi szakaszt. Ennek megfelelően az árakat az alábbi táblázat tartalmazza.

Tavasz	Nyár	Ősz
01. 01. – 04. 30.	05. 01. – 08. 31.	09. 01. – 12. 31.
9 000 Ft	10 000 Ft	8 000 Ft

A feltüntetett értékek egy szoba árát mutatják egy éjszakára. Ha csak egy fő száll meg, akkor is ki kell fizetni a teljes szobaárat. Egy adott foglalás besorolása az érkezés napjától függ.

A pótágy díja 2 000 Ft éjszakánként. Amennyiben a vendég igényel reggelit, azért a fenti áron felül személyenként és naponként 1 100 Ft-ot kell fizetni.

Ha például a két felnőttből és egy gyermekből álló Tóth család április 30. és május 4. között 4 éjszakát tölt a hotelben és kér reggelit, akkor ők az alábbi összegeket fizetik:

- 4 × 9 000 Ft-ot a szobáért,
- 4 × 2 000 Ft-ot a pótágyért,
- 4 × 3 × 1 100 Ft-ot a reggeliért.

A végső számla így 36 000 Ft + 8 000 Ft + 13 200 Ft = 57 200 Ft lesz.

A szálloda eddigi foglalásait a *pitypang.txt* fájl tartalmazza. Az első sor a fájlban tárolt foglalások számát mutatja. A további sorokban szóközzel elválasztva soronként az alábbi adatok találhatók:

- a foglalás sorszáma,
- a szoba száma (1–27),
- az érkezés napjának sorszáma,
- a távozás napjának sorszáma,
- a vendégek száma,
- kérnek-e reggelit (1=igen vagy 0=nem),
- a foglalást végző vendég nevéből képzett azonosítója (maximum 25 karakter).

A napok sorszámozása január 1-jétől (1-es sorszám) kezdődik. Április 30-hoz például a 31 + 28 + 31 + 30 = 120-as sorszám tartozik.

A Tóth család foglalása ebben a szerkezetben a következőképpen néz ki:

```
123 21 120 124 3 1 Toth_Balint
```

A fájl egy éven belül tartalmaz foglalásokat. Az adatok az érkezés napja szerint növekvő sorrendben vannak rendezve a fájlban.

Tájékoztatásul a *honapok.txt* fájl a hónapok neveit, a rá következő sorban az adott hónap napjainak számát, majd az ezt követő sorban pedig a hónap első napjának sorszámát tartalmazza.

Az állományt forrásfájlként is felhasználhatja. A fenti táblázatnak megfelelő nyári időszak a 121. napon, míg az őszi a 244. napon kezdődik.

Készítsen programot *szalloda* néven, amely az alábbi kérdésekre válaszol!

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például `3. feladat:`)! Ahol a felhasználótól kér be adatot, ott írja a képernyőre, hogy milyen adatot vár!

1. Olvassa be az *pitypang.txt* állományban található maximum 1 000 foglalás adatát, s annak felhasználásával oldja meg a következő feladatokat! Ha az állományt nem tudja beolvasni, akkor a benne található adatok közül az 1-5, 326-330 és 695-699 foglalási sorszámú sorok adatait jegyezze be a programba, s úgy oldja meg a feladatokat!
2. Jelenítse meg a képernyőn a leghosszabb szállodai tartózkodást! Csak az időtartamot vegye figyelembe, azaz nem számít, hogy hány vendég lakott az adott szobában! Az esetlegesen azonos hosszúságú tartózkodások közül bármelyiket kiválaszthatja. Az eredményt ebben a formában írja a képernyőre:

   Név (érkezési_nap_sorszáma) – eltöltött_éjszakák_száma
   például: `Nagy_Bertalan (105) - 16`
3. Számítsa ki, hogy az egyes foglalások után mennyit kell fizetnie az egyes vendégeknek! A foglalás sorszámát és a kiszámított értékeket kettősponttal elválasztva írja ki a *bevetel.txt* fájlba!
   Ez – a példában szereplő Tóth család esetén – a következő lenne:

   `123:57200`

   (Amennyiben nem tudja a fájlba íratni a kiszámított értékeket, úgy az első tíz foglaláshoz tartozó értéket a képernyőre írassa ki!)
   Írja a képernyőre a szálloda teljes évi bevételét!
4. Készítsen statisztikát az egyes hónapokban eltöltött vendégéjszakákról! Egy vendégéjszakának egy fő egy eltöltött éjszakája számít. A példában szereplő Tóth család áprilisban 3, májusban pedig 9 vendégéjszakát töltött a szállodában. Írassa ki a havi vendégéjszakák számát a képernyőre az alábbi formában:

   hónap_sorszáma: x vendégéj
   például: `8: 1059 vendégéj`
5. Kérje be a felhasználótól egy új foglalás kezdő dátumához tartozó nap sorszámát és az eltöltendő éjszakák számát! Határozza meg, hogy hány szoba szabad a megadott időszak teljes időtartamában! A választ írassa ki a képernyőre!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Pitypang
 *
 * @author Klemand
 */

public class EmeltInfo2011okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");

		System.out.println("Az adatok beolvasása a honapok.txt fájlból");
		BufferedReader beolvas = new BufferedReader(new FileReader("honapok.txt"));

		int[] honapHossz = new int[12];
		int[] honapKezdonap = new int[12];

		int i;
		for (i = 1; i <= 12; i++) {
			beolvas.readLine();
			// A hónapnév nem kell, csak átugorjuk
			honapHossz[i - 1] = Integer.parseInt(beolvas.readLine());
			honapKezdonap[i - 1] = Integer.parseInt(beolvas.readLine());
		}
		beolvas.close();

		System.out.println("Az adatok beolvasása a pitypang.txt fájlból");
		beolvas = new BufferedReader(new FileReader("pitypang.txt"));

		int db = Integer.parseInt(beolvas.readLine());

		int[] szoba = new int[db];
		int[] erkezes = new int[db];
		int[] tavozas = new int[db];
		int[] vendeg = new int[db];
		int[] reggeli = new int[db];
		String[] nev = new String[db];
		String[] daraboltSor;

		for (i = 1; i <= db; i++) {
			daraboltSor = beolvas.readLine().split(" ");
			// A foglalás számát nem tároljuk, az a tömb indexe!
			szoba[i - 1] = Integer.parseInt(daraboltSor[1]);
			erkezes[i - 1] = Integer.parseInt(daraboltSor[2]);
			tavozas[i - 1] = Integer.parseInt(daraboltSor[3]);
			vendeg[i - 1] = Integer.parseInt(daraboltSor[4]);
			reggeli[i - 1] = Integer.parseInt(daraboltSor[5]);
			nev[i - 1] = daraboltSor[6];
		}
		beolvas.close();

		System.out.println("A beolvasások megtörténtek.\n");
```

```
System.out.println("A 2. feladat megoldása");
System.out.println("A leghosszabb tartózkodás");

int leghosszabb = 1;
for (i = 2; i <= db; i++) {
        if ((tavozas[i - 1] - erkezes[i - 1]) > ((tavozas[leghosszabb - 1] - erkezes[leghosszabb - 1]))) {
                leghosszabb = i;
        }
}
System.out.println("Név (érkezési nap sorszáma) eltöltött éjszakák száma ");
System.out.println(nev[leghosszabb - 1] + " (" + erkezes[leghosszabb - 1] + ") "
                + (tavozas[leghosszabb - 1] - erkezes[leghosszabb - 1]));

System.out.println("\nA 3. feladat megoldása");
System.out.println("Az egyes foglalások utáni bevételek kiíratása a bevetel.txt fájlba,");
System.out.println("és az összbevétel meghatározása");

PrintWriter kiir = new PrintWriter(new FileWriter("bevetel.txt"));

String ideny = "";
int bevetel;
int osszbevetel = 0;

for (i = 1; i <= db; i++) {
        int napDb = tavozas[i - 1] - erkezes[i - 1];

        if (erkezes[i - 1] < honapKezdonap[4]) {
                ideny = "tavasz";
        } else if (erkezes[i - 1] < honapKezdonap[8]) {
                ideny = "nyár";
        } else {
                ideny = "ősz";
        }

        bevetel = napDb * 8000; // őszi alapár

        if (ideny == "tavasz") {
                bevetel += napDb * 1000; // tavaszi felár
        }
        if (ideny == "nyár") {
                bevetel += napDb * 2000; // nyári felár
        }

        if (vendeg[i - 1] > 2) {
                bevetel += napDb * 2000; // pótágy
        }

        if (reggeli[i - 1] == 1) {
                bevetel += napDb * vendeg[i - 1] * 1100; // reggeli
        }

        kiir.println(i + ":" + bevetel);
        osszbevetel += bevetel;
}

kiir.close();
System.out.println("A bevetel.txt fájl kiíratása befejeződött.");

System.out.println("A szálloda éves összbevétele: " + osszbevetel + " Ft.\n");
```

```
                System.out.println("A 4. feladat megoldása ");
                System.out.println("Statisztika a havi vendégéjszakákról:");

                int[] ejszakaDb = new int[12];

                for (i = 1; i <= 12; i++) {
                        ejszakaDb[i - 1] = 0;
                }

                int aktHonap;
                int j;
                for (i = 1; i <= db; i++) {
                        for (j = erkezes[i - 1]; j < tavozas[i - 1]; j++) {
                                aktHonap = 1;
                                while (aktHonap <= 12 && honapKezdonap[aktHonap - 1] + honapHossz[aktHonap - 1] - 1 < j) {
                                        aktHonap++;
                                }
                                ejszakaDb[aktHonap - 1] += vendeg[i - 1];
                        }
                }

                System.out.println("A hónap sorszáma: vendégéjek száma ");

                for (i = 1; i <= 12; i++) {
                        System.out.println(i + ": " + ejszakaDb[i - 1] + " vendégéj");
                }

                System.out.println("");

                System.out.println("Az 5. feladat megoldása");
                System.out.println("Szabad szobák a felhasználó által megadott időszakban:");

                Scanner sc = new Scanner(System.in);

                System.out.println("Kérem a foglalás adatait! Az időszak teljes egészében az idei évre essen!");
                System.out.print("Kérem az érkezés időpontjának sorszámát: ");
                int ujErkezes = sc.nextInt();
                System.out.print("Kérem az eltölteni kívánt napok számát! ");
                int ujNapok = sc.nextInt();

                int szobaDb = 27;
                boolean[] szobak = new boolean[szobaDb];
                int sz;
                int szabadDb = 0;

                System.out.println("A megadott időszakban a következő szobák szabadok: ");

                for (sz = 1; sz <= szobaDb; sz++) {
                        boolean szabad = true;
                        i = 1;
                        while (i <= db && szabad) {
                                if (szoba[i - 1] == sz) {
                                        szabad = (tavozas[i - 1] <= ujErkezes || (ujErkezes + ujNapok) <= erkezes[i - 1]);
                                }
                                i++;
                        }
                        szobak[sz - 1] = szabad;
                        if (szobak[sz - 1]) {
                                szabadDb++;
                                System.out.print(sz + " ");
                        }
                }
                System.out.println("");
                System.out.println("A szabad szobák száma: " + szabadDb);

                sc.close();
                System.out.println("");
        }

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása a honapok.txt fájlból
Az adatok beolvasása a pitypang.txt fájlból
A beolvasások megtörténtek.

A 2. feladat megoldása
A leghosszabb tartózkodás
Név (érkezési nap sorszáma) eltöltött éjszakák száma
Meszaros_Agnes (121) 17

A 3. feladat megoldása
Az egyes foglalások utáni bevételek kiíratása a bevetel.txt fájlba,
és az összbevétel meghatározása
A bevetel.txt fájl kiíratása befejeződött.
A szálloda éves összbevétele: 73053900 Ft.

A 4. feladat megoldása
Statisztika a havi vendégéjszakákról:
A hónap sorszáma: vendégéjek száma
1: 752 vendégéj
2: 1027 vendégéj
3: 986 vendégéj
4: 1061 vendégéj
5: 1096 vendégéj
6: 1137 vendégéj
7: 1143 vendégéj
8: 1008 vendégéj
9: 773 vendégéj
10: 880 vendégéj
11: 1098 vendégéj
12: 672 vendégéj

Az 5. feladat megoldása
Szabad szobák a felhasználó által megadott időszakban:
Kérem a foglalás adatait! Az időszak teljes egészében az idei évre essen!
Kérem az érkezés időpontjának sorszámát: 90
Kérem az eltölteni kívánt napok számát! 2
A megadott időszakban a következő szobák szabadok:
1 2 10 11 12 13 14 20 21 22 23 24 27
A szabad szobák száma: 13
```

A bevetel.txt szövegfájl

```
1:101000
2:33600
3:54000
4:72000
5:117000
6:36000
7:112000
8:33600
9:70700
10:101000
11:9000
12:141400
13:154000
14:89600
15:40400
16:63000
17:70700
18:60600
19:28600
20:72000
21:33600
22:81000
23:27000
24:111100
25:100800
26:50500
27:72000
28:101000
29:63000
30:60600
31:80800
32:11200
33:30300
34:121000
35:18000
36:33600
37:67200
38:108000
39:20200
40:99000
41:200200
42:80800
43:117000
44:80800
45:45000
46:90000
47:101000
48:80800
49:63000
50:90000
51:143000
52:63000
53:22400
54:22400
55:131300
56:40400
57:22000
58:33600
59:101000
60:121200
61:45000
62:45000
63:99000
64:20200
65:18000
66:9000
67:36000
68:88000
69:54000
70:44800
71:63000
72:54000
73:89600
74:36000
75:112000
76:67200
77:126000
78:81000
79:90900
80:112000
...
```

2012 május: Futár

A nagyvárosokon belül, ha csomagot gyorsan kell eljuttatni egyik helyről a másikra, akkor sokszor a legjobb választás egy kerékpáros futárszolgálat igénybevétele. A futárszolgálat a futárjainak a megtett utak alapján ad fizetést. Az egyik futár egy héten át feljegyezte fuvarjai legfontosabb adatait, és azokat eltárolta egy állományban. Az állományban az adatok rögzítése nem mindig követi az időrendi sorrendet. Azokra a napokra, amikor nem dolgozott, nincsenek adatok bejegyezve az állományba.

A fájlban legalább 10 sor van, és minden sor egy-egy út adatait tartalmazza egymástól szóközzel elválasztva. Az első adat a nap sorszáma, ami 1 és 7 közötti érték lehet. A második szám a napon belüli fuvarszám, ami 1 és 40 közötti érték lehet. Ez minden nap 1-től kezdődik, és az aznapi utolsó fuvarig egyesével növekszik. A harmadik szám az adott fuvar során megtett utat jelenti kilométerben, egészre kerekítve. Ez az érték nem lehet 30-nál nagyobb.

Például:

```
1 1 5
1 2 9
3 2 12
1 4 3
3 1 7
…
```

A 3. sor például azt mutatja, hogy a hét harmadik napján a második fuvar 12 kilométeres távolságot jelentett.

Készítsen programot, amely a *tavok.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *futar* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be a *tavok.txt* állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat!
2. Írja ki a képernyőre, hogy mekkora volt a hét legelső útja kilométerben! Figyeljen arra, hogy olyan állomány esetén is helyes értéket adjon, amiben például a hét első napján a futár nem dolgozott!
3. Írja ki a képernyőre, hogy mekkora volt a hét utolsó útja kilométerben!
4. Tudjuk, hogy a futár minden héten tart legalább egy szabadnapot. Írja ki a képernyőre, hogy a hét hányadik napjain nem dolgozott a futár!
5. Írja ki a képernyőre, hogy a hét melyik napján volt a legtöbb fuvar!
   Amennyiben több nap is azonos, maximális számú fuvar volt, elegendő ezek egyikét kiírnia.

6. Számítsa ki és írja a képernyőre a mintának megfelelően, hogy az egyes napokon hány kilométert kellett tekerni!

```
1. nap: 124 km
2. nap: 0 km
3. nap: 75 km …
```

7. A futár az egyes utakra az út hosszától függően kap fizetést az alábbi táblázatnak megfelelően:

1 – 2 km	500 Ft
3 – 5 km	700 Ft
6 – 10 km	900 Ft
11 – 20 km	1 400 Ft
21 – 30 km	2 000 Ft

Kérjen be a felhasználótól egy tetszőleges távolságot, és határozza meg, hogy mekkora díjazás jár érte! Ezt írja a képernyőre!

8. Határozza meg az összes rögzített út ellenértékét! Ezeket az értékeket írja ki a *dijazas.txt* állományba nap szerint, azon belül pedig az út sorszáma szerinti növekvő sorrendben az alábbi formátumban:

```
1. nap 1. út: 700 Ft
1. nap 2. út: 900 Ft
1. nap 3. út: 2000 Ft
…
```

9. Határozza meg, és írja ki a képernyőre, hogy a futár mekkora összeget kap a heti munkájáért!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Futár
 *
 * @author Klemand
 */

public class EmeltInfo2012maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása a tavok.txt fájlból");

		BufferedReader beolvas = new BufferedReader(new FileReader("tavok.txt"));
		int[][] ut = new int[280][3];
		/*
		 * 7 nap, napi max 40 fuvar, ez max. 280 út a nap sorszáma (1-7), a napi fuvar
		 * sorszáma (max. 40) és a fuvar távolsága (max. 30 km)
		 */

		String sor;
		String[] daraboltSor;
		int db = 0;
		int j;
		while ((sor = beolvas.readLine()) != null) {
			daraboltSor = sor.split(" ");
			db++;
			for (j = 1; j <= 3; j++) {
				ut[db - 1][j - 1] = Integer.parseInt(daraboltSor[j - 1]);
			}
		}
		beolvas.close();

		System.out.println("A beolvasás megtörtént. A fuvarok száma " + db);

		System.out.println("\nA 2. feladat megoldása ");

		System.out.println("A tömb rendezése napok és azon belül fuvarok szerint");

		// Először a másodlagos szempont szerint rendezünk!
		System.out.println("Sorszám szerinti rendezés buborékos rendezéssel");

		int i;
		int[] asztal;

		for (i = db - 1; i >= 1; i--) {
			for (j = 1; j <= i; j++) {
				if (ut[j - 1][1] > ut[j][1]) {
					asztal = ut[j - 1];
					ut[j - 1] = ut[j];
					ut[j] = asztal;
				}
			}
		}
```

```
// A főszempont szerinti rendezésnél már semmiképp sem lehetnek távoli cserék!
System.out.println("Napok szerinti rendezés buborékos rendezéssel");

for (i = db - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (ut[j - 1][0] > ut[j][0]) {
                        asztal = ut[j - 1];
                        ut[j - 1] = ut[j];
                        ut[j] = asztal;
                }
        }
}

System.out.println("A hét első útja " + ut[0][2] + " km volt. \n");

System.out.println("A 3. feladat megoldása ");
System.out.println("A hét utolsó útja " + ut[db - 1][2] + " km volt. \n");

System.out.println("A 4. feladat megoldása ");
System.out.print("Szabadnap:");

/*
 * Tudjuk, hogy volt legalább 1 szabadnap! Először a fuvarDb tömbbe tesszük a
 * hét egyes napjain teljesített fuvarok számát.
 */

int[] fuvarDb = new int[7];

for (i = 1; i <= 7; i++) {
        fuvarDb[i - 1] = 0;
}

int aktNap;
for (i = 1; i <= db; i++) {
        aktNap = ut[i - 1][0];
        fuvarDb[aktNap - 1]++;
}

// Amelyik napon 0 fuvarja volt, azon nem dolgozott
for (i = 1; i <= 7; i++) {
        if (fuvarDb[i - 1] == 0) {
                System.out.print(" " + i);
        }
}
System.out.println("\n");

System.out.println("Az 5. feladat megoldása");
System.out.print("A legtöbb fuvar a(z) ");

int maxFuvar = 1;

for (i = 2; i <= 7; i++) {
        if (fuvarDb[i - 1] > fuvarDb[maxFuvar - 1]) {
                maxFuvar = i;
        }
}

System.out.println(maxFuvar + ". napon volt: " + fuvarDb[maxFuvar - 1] + " út.");
```

```
        System.out.println("\nA 6. feladat megoldása");
        System.out.println("A teljesített km-ek száma napi bontásban: ");

        /*
         * Először meghatározzuk, és a kmDb tömbbe tesszük a hét egyes napjain
         * teljesített km-ek számát.
         */
        int[] kmDb = new int[7];

        for (i = 1; i <= 7; i++) {
                kmDb[i - 1] = 0;
        }

        for (i = 1; i <= db; i++) {
                aktNap = ut[i - 1][0];
                kmDb[aktNap - 1] += ut[i - 1][2];
        }

        for (i = 1; i <= 7; i++) {
                System.out.print(i + ". nap: ");
                System.out.printf("%3d", kmDb[i - 1]);
                System.out.println(" km");
        }

        System.out.println("\nA 7. feladat megoldása");
        System.out.println("Egy bekért hosszúságú út díjazásának meghatározása díjsávok alapján ");

        Scanner sc = new Scanner(System.in);

        System.out.print("Kérem az út hosszát km-ben: ");
        int km = sc.nextInt();

        int dijazas = dijazasMeghat(km);
        System.out.println("A fuvar díjazása: " + dijazas + " Ft");

        sc.close();

        System.out.println("\nA 8. feladat megoldása");
        System.out.println("Az összes út díjazásának kiíratása a dijazas. txt fájlba");
        System.out.println("napok és azon belül az út sorszáma szerint növekvő sorrendben ");

        PrintWriter kiir = new PrintWriter(new FileWriter("dijazas.txt"));

        int osszDijazas = 0;
        for (i = 1; i <= db; i++) {
                dijazas = dijazasMeghat(ut[i - 1][2]);
                kiir.println(ut[i - 1][0] + ". nap " + ut[i - 1][1] + ". út: " + dijazas + " Ft");
                osszDijazas += dijazas;
                // Ez a 9. feladathoz kell
        }

        kiir.close();
        System.out.println("A dijazas.txt fájl kiíratása befejeződött. \n");

        System.out.println("A 9. feladat megoldása");
        System.out.println("A futár heti munkájának díjazása: " + osszDijazas + " Ft.");
        System.out.println("");

}
```

```
        public static int dijazasMeghat(int ut) {

                if (ut < 3) {
                        return 500;
                } else if (ut < 6) {
                        return 700;
                } else if (ut < 11) {
                        return 900;
                } else if (ut < 21) {
                        return 1400;
                } else {
                        return 2000;
                }
        }

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása a tavok.txt fájlból
A beolvasás megtörtént. A fuvarok száma 61

A 2. feladat megoldása
A tömb rendezése napok és azon belül fuvarok szerint
Sorszám szerinti rendezés buborékos rendezéssel
Napok szerinti rendezés buborékos rendezéssel
A hét első útja 3 km volt.

A 3. feladat megoldása
A hét utolsó útja 25 km volt.

A 4. feladat megoldása
Szabadnap: 2 6

Az 5. feladat megoldása
A legtöbb fuvar a(z) 5. napon volt: 21 út.

A 6. feladat megoldása
A teljesített km-ek száma napi bontásban:
1. nap:  65 km
2. nap:   0 km
3. nap:  69 km
4. nap:  62 km
5. nap:  74 km
6. nap:   0 km
7. nap:  75 km

A 7. feladat megoldása
Egy bekért hosszúságú út díjazásának meghatározása díjsávok alapján
Kérem az út hosszát km-ben: 28
A fuvar díjazása: 2000 Ft

A 8. feladat megoldása
Az összes út díjazásának kiíratása a dijazas. txt fájlba
napok és azon belül az út sorszáma szerint növekvő sorrendben
A dijazas.txt fájl kiíratása befejeződött.

A 9. feladat megoldása
A futár heti munkájának díjazása: 48500 Ft.
```

A dijazas.txt szövegfájl

```
1. nap 1. út: 700 Ft
1. nap 2. út: 700 Ft
1. nap 3. út: 500 Ft
1. nap 4. út: 900 Ft
1. nap 5. út: 700 Ft
1. nap 6. út: 500 Ft
1. nap 7. út: 1400 Ft
1. nap 8. út: 900 Ft
1. nap 9. út: 900 Ft
1. nap 10. út: 700 Ft
1. nap 11. út: 900 Ft
3. nap 1. út: 900 Ft
3. nap 2. út: 900 Ft
3. nap 3. út: 900 Ft
3. nap 4. út: 500 Ft
3. nap 5. út: 700 Ft
3. nap 6. út: 900 Ft
3. nap 7. út: 700 Ft
3. nap 8. út: 700 Ft
3. nap 9. út: 900 Ft
3. nap 10. út: 500 Ft
3. nap 11. út: 700 Ft
3. nap 12. út: 500 Ft
3. nap 13. út: 700 Ft
3. nap 14. út: 700 Ft
3. nap 15. út: 500 Ft
4. nap 1. út: 1400 Ft
4. nap 2. út: 500 Ft
4. nap 3. út: 700 Ft
4. nap 4. út: 2000 Ft
4. nap 5. út: 900 Ft
4. nap 6. út: 1400 Ft
5. nap 1. út: 500 Ft
5. nap 2. út: 700 Ft
5. nap 3. út: 500 Ft
5. nap 4. út: 900 Ft
5. nap 5. út: 500 Ft
5. nap 6. út: 700 Ft
5. nap 7. út: 500 Ft
5. nap 8. út: 900 Ft
5. nap 9. út: 500 Ft
5. nap 10. út: 500 Ft
5. nap 11. út: 700 Ft
5. nap 12. út: 500 Ft
5. nap 13. út: 500 Ft
5. nap 14. út: 700 Ft
5. nap 15. út: 700 Ft
5. nap 16. út: 700 Ft
5. nap 17. út: 900 Ft
5. nap 18. út: 500 Ft
5. nap 19. út: 700 Ft
5. nap 20. út: 900 Ft
5. nap 21. út: 500 Ft
7. nap 1. út: 700 Ft
7. nap 2. út: 1400 Ft
7. nap 3. út: 900 Ft
7. nap 4. út: 900 Ft
7. nap 5. út: 900 Ft
7. nap 6. út: 900 Ft
7. nap 7. út: 900 Ft
7. nap 8. út: 2000 Ft
```

2012 május idegennyelvű: Törtek

A matematikában sokszor van szükségünk műveletvégzésre a közönséges törtekkel. A legtöbb számológép és számítógépes program csak a tizedestörteket ismeri.

Készítsen programot, amely az alábbi – közönséges törtekkel kapcsolatos – feladatokat megoldja! A program forráskódját *tort* néven mentse! A feladatban csak pozitív számokkal kell dolgoznia, és ennek a tulajdonságnak a feldolgozandó fájlban található számadatok is megfelelnek. A felhasználótól bekérendő és a feldolgozandó fájlban található számokról feltételezheti, hogy legfeljebb kétjegyűek.

Minden – képernyőre írást igénylő – részfeladat megoldása előtt írja a képernyőre a feladat sorszámát! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár (például az 1. feladat esetén: „`1. feladat Adja meg a számlálót: `”)!
Az ékezetmentes kiírás is elfogadott.

1. Kérjen be a felhasználótól két számot, amely egy közönséges tört számlálója és nevezője! Döntse el, hogy az így bevitt tört felírható-e egész számként! Ha igen, írja ki értékét egész számként, ha nem, írja ki „`Nem egész`”!
2. A közönséges törteket úgy tudjuk a legegyszerűbb alakra hozni, ha a számlálóját és nevezőjét elosztjuk a két szám legnagyobb közös osztójával, és az így kapott érték lesz az új számláló, illetve nevező. Az egyszerűsítéshez készítsen egy rekurzív függvényt az alább leírt euklideszi algoritmusnak megfelelően!

```
Függvény lnko(a, b : egész számok) : egész szám
   ha a=b akkor lnko := a
   ha a<b akkor lnko := lnko(a, b-a)
   ha a>b akkor lnko := lnko(a-b, b)
Függvény vége
```

3. Az első feladatban bekért törtet hozza a legegyszerűbb alakra a létrehozott függvény segítségével! Amennyiben nem sikerül az előírt függvényt elkészítenie, alkalmazhat más megoldást, hogy a további feladatokat meg tudja oldani. Az eredményt írja ki a következő formában:

```
24/32 = 3/4
```

   Amennyiben a tört felírható egész számként, akkor ebben az alakban jelenjen meg:

```
24/6 = 4
```

4. Két törtet úgy tudunk összeszorozni, hogy a két tört számlálóját összeszorozva kapjuk az eredmény számlálóját, és a két tört nevezőjét összeszorozva kapjuk az eredmény nevezőjét. Kérjen be a felhasználótól egy újabb közönséges törtet a számlálójával és a nevezőjével! Szorozza meg ezzel a törttel az első feladatban bekért törtet! Az eredményt hozza a legegyszerűbb alakra, és ezt írja ki a következő formában:

```
24/32 * 12/15 = 288/480 = 3/5
```

Amennyiben az eredmény felírható egész számként, akkor ebben az alakban jelenjen meg:

```
24/32 * 8/3 = 192/96 = 2
```

5. Két közönséges tört összeadásához a következő lépésekre van szükség:

   - Mindkét számot bővíteni kell, azaz mind a számlálóját, mind a nevezőjét ugyanazzal a számmal kell megszorozni. Ezt a bővítést úgy célszerű elvégezni, hogy a közös nevező a két eredeti nevező legkisebb közös többszöröse legyen. Ez lesz az összeg nevezője.
   - A két bővített alakú tört számlálóját összeadjuk, ez lesz az eredmény számlálója.

   Ehhez készítsen függvényt az alábbiakban leírtak szerint – a korábban elkészített *lnko* függvény felhasználásával – a legkisebb közös többszörös meghatározására!

```
Függvény lkkt(a, b : egész számok) : egész szám
   lkkt := a * b / lnko(a, b)
Függvény vége
```

6. A függvény segítségével határozza meg a két bekért tört összegét, és ezt adja meg a következő formában! (Amennyiben nem sikerül az előírt függvényt elkészítenie, alkalmazhat más megoldást, hogy a további feladatokat meg tudja oldani.)

```
24/32 + 8/3 = 72/96 + 256/96 = 328/96 = 41/12
```

   Amennyiben az eredmény felírható egész számként, akkor ebben az alakban jelenjen meg:

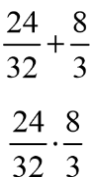
```
22/4 + 27/6 = 66/12 + 54/12 = 120/12 = 10
```

7. Az *adat.txt* állományban található műveleteket végezze el, és az eredményeket a korábbi, képernyőre kiírt formátumnak megfelelően írja az *eredmeny.txt* állományba! Az *adat.txt* fájlnak legfeljebb 100 sora lehet; soronként 4 számot és egy műveleti jelet tartalmaz, melyeket mindenhol egy szóköz választ el egymástól. Műveleti jelként csak összeadás és szorzás szerepel.

   Például:

adat.txt:

```
24 32 8 3 +
24 32 8 3 *
```



$$\frac{24}{32}+\frac{8}{3}$$

$$\frac{24}{32}\cdot\frac{8}{3}$$

eredmeny.txt:

```
24/32 + 8/3 = 72/96 + 256/96 = 328/96 = 41/12
24/32 * 8/3 = 192/96 = 2
```

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Törtek
 *
 * @author Klemand
 */

public class EmeltInfo2012mid {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Egy tört számlálójának és nevezőjének bekérése, ");
		System.out.println("és annak eldöntése, hogy a tört értéke egész szám-e");

		Scanner sc = new Scanner(System.in);

		System.out.print("Kérem a számlálót (max. 99): ");
		int sz1 = sc.nextInt();
		System.out.print("Kérem a nevezőt (max. 99): ");
		int n1 = sc.nextInt();

		if (sz1 % n1 == 0) {
			System.out.println("Egész: " + sz1 / n1);
		} else {
			System.out.println("Nem egész");
		}
		System.out.println("");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Rekurzív függvény készítése az LNKO meghatározására \n");
		// A függvény kódolása a main metódus után található

		System.out.println("A 3. feladat megoldása");
		System.out.println("A bekért tört legegyszerűbb alakra hozása az elkészített függvény segítségével");

		System.out.println(sz1 + "/" + n1 + " = " + egyszerusites(sz1, n1));
		System.out.println("");

		System.out.println("A 4. feladat megoldása");
		System.out.println("A korábban bekért tört és egy újabb bekért tört szorzása és egyszerűsítése");

		System.out.print("Kérem az újabb tört számlálóját (max. 99): ");
		int sz2 = sc.nextInt();
		System.out.print("Kérem az újabb tört nevezőjét (max. 99): ");
		int n2 = sc.nextInt();
		System.out.println("A két tört szorzata:");
		System.out.println(sz1 + "/" + n1 + " * " + sz2 + "/" + n2 + " = " + sz1 * sz2 + "/" + n1 * n2 + " = "
				+ egyszerusites(sz1 * sz2, n1 * n2));

		sc.close();

		System.out.println("\nAz 5. feladat megoldása");
		System.out.println("Függvény készítése az LKKT meghatározására \n");
		// A függvény kódolása a main metódus után található
```

```
        System.out.println("A 6. feladat megoldása");
        System.out.println("A két bekért tört összadása és egyszerűsítése:");
        System.out.println(sz1 + "/" + n1 + " + " + sz2 + "/" + n2 + " = " + sz1 * lkkt(n1, n2) / n1 + "/"
                + lkkt(n1, n2) + " + " + sz2 * lkkt(n1, n2) / n2 + "/" + lkkt(n1, n2) + " = "
                + (sz1 * lkkt(n1, n2) / n1 + sz2 * lkkt(n1, n2) / n2) + "/" + lkkt(n1, n2) + " = "
                + egyszerusites(sz1 * lkkt(n1, n2) / n1 + sz2 * lkkt(n1, n2) / n2, lkkt(n1, n2)));

        System.out.println("\nA 7. feladat megoldása");
        System.out.println("Az adat.txt fájlban lévő műveletek eredményének kiíratása az eredmeny.txt fájlba ");
        System.out.println("az előzőeknek megfelelő formában");

        /*
         * Felesleges lenne tárolni az adatokat, csupán egyszer és egyenként
         * használjuk a fájl sorait! Beolvasunk egy sort, feldolgozzuk és rögtön
         * ki is írjuk az eredményt a másik fájlba.
         */

        BufferedReader beolvas = new BufferedReader(new FileReader("adat.txt"));

        PrintWriter kiir = new PrintWriter(new FileWriter("eredmeny.txt"));

        String sor;
        String[] daraboltSor;

        while ((sor = beolvas.readLine()) != null) {
            daraboltSor = sor.trim().split(" ");
            if (daraboltSor[4].equals("*")) {
                szorzatKiiras(kiir, Integer.parseInt(daraboltSor[0]),
                                    Integer.parseInt(daraboltSor[1]),
                                    Integer.parseInt(daraboltSor[2]),
                                    Integer.parseInt(daraboltSor[3]));
            } else {
                osszegKiiras(kiir, Integer.parseInt(daraboltSor[0]),
                                    Integer.parseInt(daraboltSor[1]),
                                    Integer.parseInt(daraboltSor[2]),
                                    Integer.parseInt(daraboltSor[3]));
            }
        }

        beolvas.close();
        kiir.close();
        System.out.println("Az eredmeny.txt fájl kiíratása megtörtént. \n");

    }


    public static int lnko(int a, int b) {
        if (a == b) {
            return a;
        }
        if (a < b) {
            return lnko(a, b - a);
        } else {
            return lnko(a - b, b);
        }
    }

    public static String egyszerusites(int a, int b) {
        if (a % b == 0) {
            return Integer.toString(a / b);
        } else {
            return Integer.toString(a / lnko(a, b)) + "/" + Integer.toString(b / lnko(a, b));
        }
    }
```

```
        public static int lkkt(int a, int b) {
                return (a * b) / lnko(a, b);
        }

        public static void szorzatKiiras(PrintWriter kivitel, int sz1, int n1, int sz2, int n2) {
                kivitel.println(sz1 + "/" + n1 + " * " + sz2 + "/" + n2 + " = "
                                + sz1 * sz2 + "/" + n1 * n2 + " = "
                                + egyszerusites(sz1 * sz2, n1 * n2));
        }

        public static void osszegKiiras(PrintWriter kivitel, int sz1, int n1, int sz2, int n2) {
                kivitel.println(sz1 + "/" + n1 + " + " + sz2 + "/" + n2 + " = "
                                + sz1 * lkkt(n1, n2) / n1 + "/" + lkkt(n1, n2)
                                + " + " + sz2 * lkkt(n1, n2) / n2 + "/" + lkkt(n1, n2) + " = "
                                + (sz1 * lkkt(n1, n2) / n1 + sz2 * lkkt(n1, n2) / n2) + "/" + lkkt(n1, n2) + " = "
                                + egyszerusites(sz1 * lkkt(n1, n2) / n1 + sz2 * lkkt(n1, n2) / n2, lkkt(n1, n2)));
        }

}
```

```
Az 1. feladat megoldása
Egy tört számlálójának és nevezőjének bekérése,
és annak eldöntése, hogy a tört értéke egész szám-e
Kérem a számlálót (max. 99): 33
Kérem a nevezőt (max. 99): 90
Nem egész

A 2. feladat megoldása
Rekurzív függvény készítése az LNKO meghatározására

A 3. feladat megoldása
A bekért tört legegyszerűbb alakra hozása az elkészített függvény segítségével
33/90 = 11/30

A 4. feladat megoldása
A korábban bekért tört és egy újabb bekért tört szorzása és egyszerűsítése
Kérem az újabb tört számlálóját (max. 99): 77
Kérem az újabb tört nevezőjét (max. 99): 28
A két tört szorzata:
33/90 * 77/28 = 2541/2520 = 121/120

Az 5. feladat megoldása
Függvény készítése az LKKT meghatározására

A 6. feladat megoldása
A két bekért tört összadása és egyszerűsítése:
33/90 + 77/28 = 462/1260 + 3465/1260 = 3927/1260 = 187/60

A 7. feladat megoldása
Az adat.txt fájlban lévő műveletek eredményének kiíratása az eredmeny.txt fájlba
az előzőeknek megfelelő formában
Az eredmeny.txt fájl kiíratása megtörtént.
```

Az eredmeny.txt szövegfájl

```
30/12 + 8/8 = 60/24 + 24/24 = 84/24 = 7/2
12/6 + 20/6 = 12/6 + 20/6 = 32/6 = 16/3
12/12 + 27/6 = 12/12 + 54/12 = 66/12 = 11/2
25/30 + 10/20 = 50/60 + 30/60 = 80/60 = 4/3
4/15 + 27/20 = 16/60 + 81/60 = 97/60 = 97/60
8/20 * 8/30 = 64/600 = 8/75
30/5 + 75/30 = 180/30 + 75/30 = 255/30 = 17/2
18/2 + 9/50 = 450/50 + 9/50 = 459/50 = 459/50
12/50 + 25/6 = 36/150 + 625/150 = 661/150 = 661/150
6/8 + 12/6 = 18/24 + 48/24 = 66/24 = 11/4
30/18 + 12/15 = 150/90 + 72/90 = 222/90 = 37/15
10/4 + 12/3 = 30/12 + 48/12 = 78/12 = 13/2
6/6 * 50/12 = 300/72 = 25/6
8/6 * 6/4 = 48/24 = 2
10/4 * 45/4 = 450/16 = 225/8
2/75 * 75/8 = 150/600 = 1/4
8/4 * 4/30 = 32/120 = 4/15
8/12 + 30/6 = 8/12 + 60/12 = 68/12 = 17/3
3/12 + 10/50 = 75/300 + 60/300 = 135/300 = 9/20
15/10 + 9/10 = 15/10 + 9/10 = 24/10 = 12/5
30/20 * 8/30 = 240/600 = 2/5
5/20 + 4/12 = 15/60 + 20/60 = 35/60 = 7/12
2/6 + 20/20 = 20/60 + 60/60 = 80/60 = 4/3
25/10 * 10/30 = 250/300 = 5/6
12/5 * 45/10 = 540/50 = 54/5
2/4 * 10/12 = 20/48 = 5/12
...
```

2012 október: Szín-kép

Egy digitális kép tárolásánál minden egyes képpont színét tároljuk. A képpontok színét az RGB kód adja. Az RGB kód a vörös (R), zöld (G) és a kék (B) színösszetevő értékét határozza meg. Ezen színösszetevők értéke 0 és 255 közötti egész szám lehet.

A *kep.txt* fájlban egy 50×50 képpontos kép képpontjainak RGB kódjai vannak a következő formában. Az állomány a képet sorfolytonosan, a képpontok RGB kódját szóközzel elválasztva tartalmazza, minden képpontot egy újabb sorban:

```
200 96 64
200 96 64
200 96 64
200 96 64
200 96 64
```

Készítsen programot *szinkep* néven a következő feladatok megoldására! A program futása során a képernyőre való kiíráskor, illetve az adatok billentyűzetről való beolvasásakor utaljon a feladat sorszámára és a kiírandó, illetve bekérendő adatra!

1. Olvassa be a fájlból egy megfelelő adatszerkezetbe az egyes képpontok RGB kódját!
2. Kérjen be a felhasználótól egy RGB kódot! Állapítsa meg a program segítségével, hogy a bekért szín megtalálható-e a képen! A megállapítás eredményét írja ki a képernyőre!
3. Határozza meg, hogy a kép 35. sor 8. képpontjának színe hányszor szerepel a 35. sorban, illetve a 8. oszlopban. Az értékeket írja ki a képernyőre az alábbi formában:

   Például:

   ```
   Sorban: 5 Oszlopban: 10
   ```
4. Állapítsa meg, hogy a vörös, kék és zöld színek közül melyik szín fordul elő legtöbbször a képen! Az (egyik) legtöbbször előforduló szín nevét írja ki a képernyőre!

   A színek kódjai:

Vörös	255, 0, 0
Zöld	0, 255, 0
Kék	0, 0, 255

5. Készítsen 3 képpont széles, fekete színű keretet a képnek! A keretet úgy hozza létre, hogy a kép mérete ne változzon! A fekete szín kódja RGB (0, 0, 0).

6. A kép képpontjainak színét írja ki a *keretes.txt* nevű szövegfájlba a bemeneti fájl formátumával egyezően! A képet sorfolytonosan tárolja, minden képpontot új sorba, a képpontok RGB kódját szóközzel elválasztva írja ki!

   Például:

```
...
0 0 0
0 0 0
200 96 64 ...
```

7. Az 50×50-es képen a kerettől függetlenül egy sárga RGB (255, 255, 0) színű téglalap van. Határozza meg a program segítségével a bal felső és a jobb alsó sárga képpontnak a helyét (sor, oszlop), majd határozza meg, hogy a sárga téglalap hány képpontból áll! A képpontok helyét és a sárga alakzat méretét a következő formában írassa ki a képernyőre:

```
Kezd: sor, oszlop
Vége: sor, oszlop
Képpontok száma: darab
```

   Például:

```
Kezd: 18, 12
Vége: 25, 19
Képpontok száma: 64
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Szín-kép
 *
 * @author Klemand
 */

public class EmeltInfo2012okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A kep.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("kep.txt"));

		String[][] kep = new String[50][50]; // sor, oszlop

		/*
		 * Tudom, hogy a kép 50x50 képpontos, így a fájl 2500 sorból áll.
		 * Sorfolytonos ábrázolás: külső ciklus a sor, belső az oszlop,
		 * először az első sor, majd a második, stb.
		 */

		int i, j;
		for (i = 1; i <= 50; i++) {
			for (j = 1; j <= 50; j++) {
				kep[i - 1][j - 1] = beolvas.readLine();
			}
		}

		beolvas.close();

		System.out.println("A kép ábrázolása (nem része a feladatnak)");
		System.out.println("A beolvasott kép vörös, zöld, kék, sárga és fekete");
		System.out.println("pontjainak ábrázolása színmátrixban");
		System.out.println("Vörös: V, zöld: Z, kék: K, sárga: S, magenta: M, fekete: F, egyéb szín: - ");

		for (i = 1; i <= 50; i++) {
			for (j = 1; j <= 50; j++) {
				System.out.print(szinjel(kep[i - 1][j - 1]));
			}
			System.out.println("");
		}
		System.out.println("");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Egy szín keresése a képen bekért RGB kód alapján");

		Scanner sc = new Scanner(System.in);

		System.out.print("Kérem az RGB kód R (vörös) összetevőjét: ");
		int r = sc.nextInt();
		System.out.print("Kérem az RGB kód G (zöld) összetevőjét: ");
		int g = sc.nextInt();
		System.out.print("Kérem az RGB kód B (kék) összetevőjét: ");
		int b = sc.nextInt();
		String kod = "" + r + " " + g + " " + b;
```

```
boolean megtalalhato = false;
int s = -1;
int o = -1;
i = 1;
while (i <= 50 && !megtalalhato) {
        j = 1;
        while (j <= 50 && !(kep[i - 1][j - 1].equals(kod))) {
                j++;
        }
        if (j <= 50) {
                megtalalhato = true;
                s = i;
                o = j;
        }
        i++;
}

if (megtalalhato) {
        System.out.println("A megadott " + kod + " szín megtalálható a képen, először a(z) "
                                                + s + ". sor " + o + ". oszlopában");
} else {
        System.out.println("A megadott " + kod + " szín nincs a képen.");

}

sc.close();

System.out.println("");

System.out.println("A 3. feladat megoldása");
System.out.print("A 35. sor 8. képpontjának színe: ");
s = 35;
o = 8;
kod = kep[s - 1][o - 1];

int sDb = 0;
for (j = 1; j <= 50; j++) {
        if (kep[s - 1][j - 1].equals(kod)) {
                sDb++;
        }
}

int oDb = 0;
for (i = 1; i <= 50; i++) {
        if (kep[i - 1][o - 1].equals(kod)) {
                oDb++;
        }
}

System.out.println(kod + ", a szín előfordulása: ");
System.out.println("Sorban: " + sDb + " Oszlopban: " + oDb);
```

```
System.out.println("\nA 4. feladat megoldása");
System.out.println("A vörös, zöld és a kék színek előfordulásainak összehasonlítása");

int vDb = 0;
int zDb = 0;
int kDb = 0;

for (i = 1; i <= 50; i++) {
        for (j = 1; j <= 50; j++) {
                kod = kep[i - 1][j - 1];

                switch (kod) {
                case "255 0 0":
                        vDb++;
                        break;
                case "0 255 0":
                        zDb++;
                        break;
                case "0 0 255":
                        kDb++;
                }
        }
}

System.out.print("A(z) egyik legtöbbször előforduló szín: ");

int maxDb = vDb;
String maxSzin = "vörös";
if (zDb > maxDb) {
        maxDb = zDb;
        maxSzin = "zöld";
}
if (kDb > maxDb) {
        maxDb = kDb;
        maxSzin = "kék";
}
System.out.println(maxSzin + " (" + maxDb + " alkalommal)\n");

System.out.println("Az 5. feladat megoldása");
System.out.println("3 képpont szélességű fekete belső keret készítése a képhez");

for (i = 1; i <= 50; i++) {
        for (j = 1; j <= 50; j++) {
                if (i <= 3 || i >= 48 || j <= 3 || j >= 48) {
                        kep[i - 1][j - 1] = "0 0 0";
                }
        }
}

System.out.println("A keretes kép ábrázolása (nem része a feladatnak)");

for (i = 1; i <= 50; i++) {
        for (j = 1; j <= 50; j++) {
                System.out.print(szinjel(kep[i - 1][j - 1]));
        }
        System.out.println("");
}
System.out.println("");
```

```
        System.out.println("A 6. feladat megoldása");
        System.out.println("A  keretes kép kiíratása a keretes.txt fájlba");

        PrintWriter kivitel;
        kivitel = new PrintWriter(new FileWriter("keretes.txt"));

        for (i = 1; i <= 50; i++) {
                for (j = 1; j <= 50; j++) {
                        kivitel.println(kep[i - 1][j - 1]);
                }
        }

        kivitel.close();
        System.out.println("A keretes.txt fájl kiíratása megtörtént. \n");

        System.out.println("A 7. feladat megoldása");
        System.out.println("A sárga téglalap adatai");

        // Tudjuk, hogy van sárga téglalap

        System.out.println("A sárga téglalap bal felső csúcsának keresése");

        boolean megvan = false;
        int s1 = -1;
        int o1 = -1;

        i = 1;
        while (i <= 50 && !megvan) {
                j = 1;
                while (j <= 50 && !(kep[i - 1][j - 1].equals("255 255 0"))) {
                        j++;
                }
                if (j <= 50) {
                        megvan = true;
                        s1 = i;
                        o1 = j;
                }
                i++;
        }

        System.out.println("Kezd: " + s1 + ", " + o1);

        System.out.println("A sárga téglalap bal felső csúcsának keresése");

        megvan = false;
        int s2 = -1;
        int o2 = -1;
        i = 50;
        while (i >= 1 && !megvan) {
                j = 50;
                while (j >= 1 && !(kep[i - 1][j - 1].equals("255 255 0"))) {
                        j--;
                }
                if (j >= 1) {
                        megvan = true;
                        s2 = i;
                        o2 = j;
                }
                i--;
        }
        System.out.println("Vége: " + s2 + ", " + o2);

        System.out.println("Képpontok száma: " + (s2 - s1 + 1) * (o2 - o1 + 1));

}
```

```
        public static String szinjel(String szin) {
                switch (szin) {
                case "255 0 0":
                        return "V";
                case "0 255 0":
                        return "Z";
                case "0 0 255":
                        return "K";
                case "255 255 0":
                        return "S";
                case "255 0 255":
                        return "M";
                case "0 0 0":
                        return "F";
                default:
                        return "-";
                }
        }

}
```

```
Az 1. feladat megoldása
A kep.txt fájl beolvasása
A kép ábrázolása (nem része a feladatnak)
A beolvasott kép vörös, zöld, kék, sárga és fekete
pontjainak ábrázolása színmátrixban
Vörös: V, zöld: Z, kék: K, sárga: S, magenta: M, fekete: F, egyéb szín: -
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
----------------------------MMMMMMMMMM----------
----------------------------MMMMMMMMMM----------
---------VVVVVVVVVV----------MMMMMMMMMM----------
---------VVVVVVVVVV------------------------------
---------VVVVVVVVVV------------------------------
---------VVVVVVVVVV------------------------------
---------VVVVVVVVVV------------------------------
---------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ----------
---------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ----------
---------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ----------
---------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ----------
---------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ----------
---------VVVVVZZZZZZZZZZZZZZZZKKKKKKKKKK----------
--------------ZZZZZZZZZZZZZZZZKKKKKKKKKK----------
--------------ZZZZZZZZZZZZZZZZKKKKKKKKKK----------
--------------ZZZZZZZZZZZZZZZZKKKKKKKKKK----------
--------------ZZZZZZZZZZZZZZZZKKKKKKKKKK----------
--------------ZZZZZZZZZZZZZZZZKKKKKKKKKK----------
----------------------------KKKKKKKKKK----------
----------------------------KKKKKKKKKK----------
----------------------------KKKKKKKKKK----------
----------------------------KKKKKKKKKK----------
----------------------------KKKKKKKKKK----------
----------SSSSSSSSS--------KKKKKKKKKK----------
----------SSSSSSSSS--------KKKKKKKKKK----------
----------SSSSSSSSS--------KKKKKKKKKK----------
----------SSSSSSSSS--------KKKKKKKKKK----------
----------SSSSSSSSS--------KKKKKKKKKK----------
----------SSSSSSSSS--------KKKKKKKKKK----------
----------SSSSSSSSS--------KKKKKKKKKK----------
----------SSSSSSSSS--------KKKKKKKKKK----------
----------SSSSSSSSS--------KKKKKKKKKK----------
----------SSSSSSSSS----------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------

A 2. feladat megoldása
Egy szín keresése a képen bekért RGB kód alapján
Kérem az RGB kód R (vörös) összetevőjét: 255
Kérem az RGB kód G (zöld) összetevőjét: 255
Kérem az RGB kód B (kék) összetevőjét: 0
A megadott 255 255 0 szín megtalálható a képen, először a(z) 31. sor 12. oszlopában

A 3. feladat megoldása
A 35. sor 8. képpontjának színe: 200 96 64, a szín előfordulása:
Sorban: 29 Oszlopban: 50

A 4. feladat megoldása
A vörös, zöld és a kék színek előfordulásainak összehasonlítása
A(z) egyik legtöbbször előforduló szín: kék (231 alkalommal)
```

```
Az 5. feladat megoldása
3 képpont szélességű fekete belső keret készítése a képhez
A keretes kép ábrázolása (nem része a feladatnak)
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFF--------------------------------------------FFF
FFF--------------------------------------------FFF
FFF--------------------------------------------FFF
FFF--------------------------------------------FFF
FFF---------------------------MMMMMMMMMMM------FFF
FFF---------------------------MMMMMMMMMMM------FFF
FFF------VVVVVVVVVVV----------MMMMMMMMMMM------FFF
FFF------VVVVVVVVVVV---------------------------FFF
FFF------VVVVVVVVVVV---------------------------FFF
FFF------VVVVVVVVVVV---------------------------FFF
FFF------VVVVVVVVVVV---------------------------FFF
FFF------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ-------FFF
FFF------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ-------FFF
FFF------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ-------FFF
FFF------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ-------FFF
FFF------VVVVVZZZZZZZZZZZZZZZZZZZZZZZZZZ-------FFF
FFF------VVVVVZZZZZZZZZZZZZZZKKKKKKKKKKK-------FFF
FFF-----------ZZZZZZZZZZZZZZZKKKKKKKKKKK-------FFF
FFF-----------ZZZZZZZZZZZZZZZKKKKKKKKKKK-------FFF
FFF-----------ZZZZZZZZZZZZZZZKKKKKKKKKKK-------FFF
FFF-----------ZZZZZZZZZZZZZZZKKKKKKKKKKK-------FFF
FFF-----------ZZZZZZZZZZZZZZZKKKKKKKKKKK-------FFF
FFF--------------------------KKKKKKKKKKK-------FFF
FFF--------------------------KKKKKKKKKKK-------FFF
FFF--------------------------KKKKKKKKKKK-------FFF
FFF--------------------------KKKKKKKKKKK-------FFF
FFF--------------------------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------KKKKKKKKKKK-------FFF
FFF--------SSSSSSSSSS--------------------------FFF
FFF--------------------------------------------FFF
FFF--------------------------------------------FFF
FFF--------------------------------------------FFF
FFF--------------------------------------------FFF
FFF--------------------------------------------FFF
FFF--------------------------------------------FFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

A 6. feladat megoldása
A  keretes kép kiíratása a keretes.txt fájlba
A keretes.txt fájl kiíratása megtörtént.

A 7. feladat megoldása
A sárga téglalap adatai
A sárga téglalap bal felső csúcsának keresése
Kezd: 31, 12
A sárga téglalap bal felső csúcsának keresése
Vége: 41, 21
Képpontok száma: 110
```

```
A keretes.txt szövegfájl

0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
...
```

2013 május: Választások

Eszemiszom városában időközi helyhatósági választásokat írtak ki. A városban összesen 12 345 szavazásra jogosult állampolgár van, akiket nyolc választókerületbe soroltak.

Minden választókerületben több jelölt is indul, de egy jelölt csak egy választókerületben indulhat. Egy választókerület szavazói az adott választókerületben induló jelöltek közül egy jelöltre adhatnak le szavazatot, de nem kötelező részt venniük a szavazáson. Minden választókerületben az a jelölt nyer, aki a legtöbb szavazatot kapja. (Feltételezheti, hogy egyetlen választókerületben sem alakult ki holtverseny.)

A jelöltek vagy egy párt támogatásával, vagy független jelöltként indulhatnak. Az idei évben a Gyümölcsevők Pártja (GYEP), a Húsevők Pártja (HEP), a Tejivók Szövetsége (TISZ) vagy a Zöldségevők Pártja (ZEP) támogatja a jelölteket.

A szavazás eredményét a *szavazatok.txt* szóközökkel tagolt fájl tartalmazza, amelynek minden sorában egy-egy képviselőjelölt adatai láthatók.

Például:

```
8 149 Zeller Zelma ZEP
6 63 Zsoldos Zsolt -
```

Az első két adat a választókerület sorszáma és a képviselőjelöltre leadott szavazatok száma. Ezt a jelölt vezeték- és utóneve, majd a jelöltet támogató párt hivatalos rövidítése követi. Független jelöltek esetében a párt rövidítése helyett egy kötőjel szerepel. Minden képviselőjelöltnek pontosan egy utóneve van.

Készítsen programot *valasztas* néven, amely az alábbi kérdésekre válaszol!

Minden részfeladat feldolgozása során írja ki a képernyőre a részfeladat sorszámát, (például: `2. feladat`)! Ahol a felhasználótól kér be adatot, ott írja ki a képernyőre azt is, hogy milyen adatot vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be a *szavazatok.txt* fájl adatait, majd ezek felhasználásával oldja meg a következő feladatokat! Az adatfájlban legfeljebb 100 képviselőjelölt adatai szerepelnek.
2. Hány képviselőjelölt indult a helyhatósági választáson? A kérdésre egész mondatban válaszoljon az alábbi mintához hasonlóan:
   `A helyhatósági választáson 92 képviselőjelölt indult.`
3. Kérje be egy képviselőjelölt vezetéknevét és utónevét, majd írja ki a képernyőre, hogy az illető hány szavazatot kapott! Ha a beolvasott név nem szerepel a nyilvántartásban, úgy jelenjen meg a képernyőn az „`Ilyen nevű képviselőjelölt nem szerepel a nyilvántartásban!`” figyelmeztetés! A feladat megoldása során feltételezheti, hogy nem indult két azonos nevű képviselőjelölt a választáson.
4. Határozza meg, hányan adták le szavazatukat, és mennyi volt a részvételi arány! (A részvételi arány azt adja meg, hogy a jogosultak hány százaléka vett részt a szavazáson.) A részvételi arányt két tizedesjegy pontossággal, százalékos formában írja ki a képernyőre!

   Például: „`A választáson 5001 állampolgár, a jogosultak 40,51%-a vett részt.`”

5. Határozza meg és írassa ki a képernyőre az egyes pártokra leadott szavazatok arányát az összes leadott szavazathoz képest két tizedesjegy pontossággal! A független jelölteket együtt, „Független jelöltek” néven szerepeltesse!

   Például:

```
Zöldségevők Pártja= 12,34%
Független jelöltek= 23,40%
```

6. Melyik jelölt kapta a legtöbb szavazatot? Jelenítse meg a képernyőn a képviselő vezeték- és utónevét, valamint az őt támogató párt rövidítését, vagy azt, hogy `független`! Ha több ilyen képviselő is van, akkor mindegyik adatai jelenjenek meg!

7. Határozza meg, hogy az egyes választókerületekben kik lettek a képviselők! Írja ki a választókerület sorszámát, a győztes vezeték- és utónevét, valamint az őt támogató párt rövidítését, vagy azt, hogy `független` egy-egy szóközzel elválasztva a *kepviselok.txt* nevű szöveges fájlba! Az adatok a választókerületek száma szerinti sorrendben jelenjenek meg!
   Minden sorba egy képviselő adatai kerüljenek!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Választások
 *
 * @author Klemand
 */

public class EmeltInfo2013maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A szavazatok.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("szavazatok.txt"));

		int[] ker = new int[100];
		int[] szavazat = new int[100];
		String[] jelolt = new String[100];
		String[] part = new String[100];
		int db = 0;
		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			daraboltSor = sor.split(" ");
			ker[db - 1] = Integer.parseInt(daraboltSor[0]);
			szavazat[db - 1] = Integer.parseInt(daraboltSor[1]);
			jelolt[db - 1] = daraboltSor[2] + " " + daraboltSor[3];
			part[db - 1] = daraboltSor[4];
			if (part[db - 1].equals("-")) {
				part[db - 1] = "független";
			}

		}
		beolvas.close();

		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("A képviselőjelöltek száma ");
		System.out.println("A választáson " + db + " képviselőjelölt indult. \n");

		System.out.println("A 3. feladat megoldása");
		System.out.println("Egy bekért képviselőjelölt szavazatainak száma");

		Scanner sc = new Scanner(System.in);

		System.out.print("Kérem a képviselőjelölt vezetéknevét: ");
		String vezNev = sc.nextLine();
		System.out.print("Kérem a képviselőjelölt utónevét: ");
		String utNev = sc.nextLine();
		String nev = vezNev + " " + utNev;
```

```
int i = 1;
while (i <= db && !(jelolt[i - 1].equals(nev))) {
        i++;
}

if (i <= db) {
        System.out.print(nev + " ");
        System.out.println(szavazat[i - 1] + " szavazatot kapott.");
} else {
        System.out.println("Ilyen nevű képviselőjelölt nem szerepel a nyilvántartásban.");
}

sc.close();

System.out.println("\nA 4. feladat megoldása");
System.out.println("A részvételi arány meghatározása");

int szavDb = 0;
for (i = 1; i <= db; i++) {
        szavDb += szavazat[i - 1];
}

System.out.print("A választáson " + szavDb + " állampolgár, a szavazásra jogosultak ");

int jogosult = 12345;
double szazalek;
szazalek = ((double) szavDb / jogosult) * 100;
System.out.printf("%.2f", szazalek);
System.out.println("%-a vett részt. \n");

System.out.println("Az 5. feladat megoldása");
System.out.println("Az egyes pártokra szavazók arányának meghatározása");

String[] partLista = new String[5];
int[] partSzav = new int[5];
int partListaDb = 0;
int j;
String aktPart;

for (i = 1; i <= db; i++) {
        aktPart = part[i - 1];
        j = 1;
        while (j <= partListaDb && !partLista[j - 1].equals(aktPart)) {
                j++;
        }
        if (j <= partListaDb) {
                partSzav[j - 1] += szavazat[i - 1];
        } else {
                partListaDb++;
                partLista[partListaDb - 1] = aktPart;
                partSzav[partListaDb - 1] = szavazat[i - 1];
        }
}

for (i = 1; i <= partListaDb; i++) {
        System.out.print(nevMeghat(partLista[i - 1]) + "= ");
        System.out.printf("%.2f", ((double) partSzav[i - 1] / szavDb) * 100);
        System.out.println("%");
}
```

```
                System.out.println("\nA 6. feladat megoldása");
                System.out.println("A legtöbb szavazatot kapott jelölt(ek)");

                int maxSzav = -1; // irreálisan kis kezdőérték
                for (i = 1; i <= db; i++) {
                        if (szavazat[i - 1] > maxSzav) {
                                maxSzav = szavazat[i - 1];
                        }
                }

                for (i = 1; i <= db; i++) {
                        if (szavazat[i - 1] == maxSzav) {
                                System.out.println(jelolt[i - 1] + " " + part[i - 1]);
                        }
                }

                System.out.println("\nA 7. feladat megoldása");
                System.out.println("A képviselők kiíratása választókerületenként a kepviselok.txt fájlba");

                PrintWriter kiir = new PrintWriter(new FileWriter("kepviselok.txt"));

                int valKerDb = 8;
                int max = -1;// kötelező kezdőérték, biztosan felülírjuk
                for (i = 1; i <= valKerDb; i++) {
                        maxSzav = -1; // irreálisan kis kezdőérték
                        for (j = 1; j <= db; j++) {
                                if ((ker[j - 1] == i) && (szavazat[j - 1] > maxSzav)) {
                                        maxSzav = szavazat[j - 1];
                                        max = j;
                                }
                        }
                        kiir.println(i + " " + jelolt[max - 1] + " " + part[max - 1]);
                }
                kiir.close();
                System.out.println("A kepviselok.txt fájl kiíratása megtörtént. \n");

        }

        public static String nevMeghat(String part) {
                switch (part) {
                case "GYEP":
                        return "Gyümölcsevők Pártja";
                case "HEP":
                        return "Húsevők Pártja";
                case "TISZ":
                        return "Tejivók Szövetsége";
                case "ZEP":
                        return "Zöldségevők Pártja";
                default:
                        return "Független jelöltek";
                }
        }

}
```

```
Az 1. feladat megoldása
A szavazatok.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
A képviselőjelöltek száma
A választáson 40 képviselőjelölt indult.

A 3. feladat megoldása
Egy bekért képviselőjelölt szavazatainak száma
Kérem a képviselőjelölt vezetéknevét: Szilva
Kérem a képviselőjelölt utónevét: Szilvia
Szilva Szilvia 87 szavazatot kapott.

A 4. feladat megoldása
A részvételi arány meghatározása
A választáson 4713 állampolgár, a szavazásra jogosultak 38,18%-a vett részt.

Az 5. feladat megoldása
Az egyes pártokra szavazók arányának meghatározása
Független jelöltek= 17,53%
Gyümölcsevők Pártja= 16,36%
Zöldségevők Pártja= 20,03%
Húsevők Pártja= 24,59%
Tejivók Szövetsége= 21,49%

A 6. feladat megoldása
A legtöbb szavazatot kapott jelölt(ek)
Joghurt Jakab TISZ
Narancs Edmond GYEP
Vadas Marcell HEP

A 7. feladat megoldása
A képviselők kiíratása választókerületenként a kepviselok.txt fájlba
A kepviselok.txt fájl kiíratása megtörtént.
```

A kepviselok.txt szövegfájl

```
1 Petrezselyem Petra ZEP
2 Oldalas Olga HEP
3 Tejes Attila TISZ
4 Monitor Tibor független
5 Joghurt Jakab TISZ
6 Vadas Marcell HEP
7 Bab Zsuzsanna ZEP
8 Narancs Edmond GYEP
```

2013 május idegennyelvű: Számok

A *Szereti Ön a számokat?* internetes vetélkedőben a versenyzők olyan kérdéseket kapnak, amelyekre egy egész számmal kell válaszolniuk. A kérdések különböző témakörökből származnak (pl. magyar, matematika, történelem, kémia), és nehézségüktől függően 1-től 3-ig terjedő pontszámot érnek. Tudjuk, hogy a kérdésekre adható válaszok értéke 0 és 1 milliárd közé esik.

A feladatokat a verseny szervezői egy adatfájlban tárolják. A fájlban minden feladat két sorban helyezkedik el. Az első sor tartalmazza a kérdést, a második pedig – egy-egy szóközzel elválasztva – a helyes választ, a helyes válaszért adható pontszámot és a témakör megnevezését. A fájlban egyelőre ékezetes betűk nem szerepelnek, pl. a „gyümölcsízű” szó helyett a „gyumolcsizu” szót írták be.

Például:

```
Mikor volt a mohacsi vesz?
1526 1 tortenelem
```

A példában szereplő kérdés: `Mikor volt a mohacsi vesz?` A helyes válasz: `1526`. A helyes válasz `1` pontot ér, és a kérdés a `tortenelem` témakörbe tartozik.

Az adatfájl még csak részben készült el. Az Ön feladata ennek a félkész adatfájlnak a tesztelése. A fájl legfeljebb 100 kérdést tartalmaz. Biztosan van benne matematika, történelem és földrajz feladat, de más témakörök is előfordulnak.

Készítsen programot, amely a *`felszam.txt`* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *`szamok`* néven! (A beolvasott fájl adatait és a felhasználó válaszainak az érvényességét nem kell ellenőriznie.)

A képernyőre írást igénylő feladatok eredményének megjelenítése előtt írja ki a képernyőre a feladat sorszámát (például: `3. feladat`)! Ha a felhasználótól kér be adatot, akkor jelenítse meg a képernyőn azt is, hogy milyen adatot vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be a *`felszam.txt`* állományban talált adatokat, és azok felhasználásával oldja meg a következő feladatokat!
2. Hány feladat van az adatfájlban? A választ írassa ki a képernyőre!
3. Határozza meg, hogy hány matematika feladat van az adatfájlban, és ezek közül hány feladat ér 1, 2, illetve 3 pontot! A választ egész mondatban írassa ki a képernyőre!

   Például:

   ```
   Az adatfajlban 20 matematika feladat van, 1 pontot er
   10 feladat, 2 pontot er 6 feladat, 3 pontot er 4 feladat.
   ```
4. Mettől meddig terjed a fájlban található válaszok számértéke? A választ egész mondatban írja ki a képernyőre!
5. Milyen témakörök szerepelnek ténylegesen az adatfájlban? Írassa ki a témakörök nevét a képernyőre úgy, hogy minden előforduló témakör pontosan egyszer jelenjen meg!

6. Kérje be egy témakör nevét, és véletlenszerűen sorsoljon ki egy kérdést ebből a témakörből! Sorsoláskor ügyeljen arra, hogy az adott témakörbe eső valamennyi feladatnak legyen esélye! (Feltételezheti, hogy a felhasználó helyesen adta meg egy létező témakör nevét.) Írassa ki a kérdést, kérje be a felhasználó válaszát, majd adja meg a válaszért járó pontszámot! (Helytelen válaszért 0 pont jár.) Ha a válasz helytelen volt, a helyes választ is közölje! A párbeszéd az alábbi formában jelenjen meg:

   Például:

```
Milyen temakorbol szeretne kerdest kapni? tortenelem
Mikor volt a mohacsi vesz? 1514
A valasz 0 pontot er.
A helyes valasz: 1526
```

7. Generáljon egy 10 kérdésből álló feladatsort véletlenszerűen úgy, hogy egyetlen feladat se szerepeljen benne kétszer! (Ügyeljen azonban arra, hogy minden beolvasott feladatnak legyen esélye a kiválasztásra!) A feladatsort írassa ki a *tesztfel.txt* állományba az alábbi formátumban! (Az első szám a helyes megoldásért járó pontszám, ezt követi a helyes válasz, majd a kérdés egy-egy szóközzel elválasztva.) Az állomány végére írassa ki a feladatsorra összesen adható pontszámot is!

   Például:

```
…
1 1526 Mikor volt a mohacsi vesz?
…
A feladatsorra osszesen 20 pont adhato.
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Számok
 *
 * @author Klemand
 */

public class EmeltInfo2013mid {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A felszam.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("felszam.txt"));

		String[] kerdes = new String[100];
		int[] valasz = new int[100];
		int[] pont = new int[100];
		String[] tema = new String[100];

		int db = 0;
		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			// csak a sorpár első tagját kell vizsgálni
			daraboltSor = beolvas.readLine().split(" ");
			db++;
			kerdes[db - 1] = sor;
			valasz[db - 1] = Integer.parseInt(daraboltSor[0]);
			pont[db - 1] = Integer.parseInt(daraboltSor[1]);
			tema[db - 1] = daraboltSor[2];
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént. \n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Az adatfájlban " + db + " feladat van. \n");

		System.out.println("A 3. feladat megoldása ");
		System.out.println("A matematika feladatok jellemzése");

		int matDb = 0;
		int[] matPontDb = new int[3];

		int i;
		for (i = 1; i <= 3; i++) {
			matPontDb[i - 1] = 0; // kezdőértékek
		}

		int aktPontszam;
		for (i = 1; i <= db; i++) {
			if (tema[i - 1].equals("matematika")) {
				matDb++;
				aktPontszam = pont[i - 1];
				matPontDb[aktPontszam - 1]++;
			}
		}
```

```
System.out.println("Az adatfájlban " + matDb + " matematika feladat van, ");
for (i = 1; i <= 3; i++) {
        System.out.print(i + " pontot ér " + matPontDb[i - 1] + " feladat");
        if (i < 3) {
                System.out.print(", ");
        } else {
                System.out.println(".\n");
        }
}

System.out.println("A 4. feladat megoldása");
System.out.println("A válaszok számértékének terjedelme");

int min = 1;
int max = 1;
for (i = 2; i <= db; i++) {
        if (valasz[i - 1] < valasz[min - 1]) {
                min = i;
        }
        if (valasz[i - 1] > valasz[max - 1]) {
                max = i;
        }
}

System.out.print("A legkisebb: " + valasz[min - 1]);
System.out.println(", a legnagyobb: " + valasz[max - 1] + ". \n");

System.out.println("Az 5. feladat megoldása");
System.out.println("Az adatfájlban szereplő témakörök:");

String[] temaLista = new String[100];
int[] temaDb = new int[100];
int temaListaDb = 0;
String aktTema;
int j;
for (i = 1; i <= db; i++) {
        aktTema = tema[i - 1];
        j = 1;
        while (j <= temaListaDb && !temaLista[j - 1].equals(aktTema)) {
                j++;
        }
        if (j <= temaListaDb) {
                temaDb[j - 1]++;
        } else {
                temaListaDb++;
                temaLista[temaListaDb - 1] = aktTema;
                temaDb[temaListaDb - 1] = 1;
        }
}

for (i = 1; i <= temaListaDb; i++) {
        System.out.println(temaLista[i - 1]);
}

System.out.println("\nA 6. feladat megoldása ");
System.out.println("Témakör bekérése - kérdés sorsolása - felelet bekérése - értékelés");

String temakor;
System.out.print("Melyik témakörből szeretne kérdést kapni? Írja be ékezet nélkül: ");

Scanner sc1 = new Scanner(System.in);
temakor = sc1.nextLine();

// Kiválasztjuk a beírt témakört, tudjuk, hogy létezik
i = 1;
while (!(temaLista[i - 1].equals(temakor))) {
        i++;
}
int aktDb = temaDb[i - 1];
```

```
                // Előállítunk egy véletlenszámot a témakör kérdéseinek sorszámai közül
                int sorszam = (int) ((Math.random() * aktDb) + 1);

                // Megkeressük az adott témakör megfelelő sorszámú kérdését
                aktDb = 0;
                i = 0;
                do {
                        i++;
                        if (tema[i - 1].equals(temakor)) {
                                aktDb++;
                        }
                } while (aktDb < sorszam);

                Scanner sc2 = new Scanner(System.in);

                System.out.print(kerdes[i - 1] + " ");
                int aktValasz = sc2.nextInt();

                if (valasz[i - 1] == aktValasz) {
                        System.out.println("A válasz " + pont[i - 1] + " pontot ér.");
                } else {
                        System.out.println("A válasz " + 0 + " pontot ér.");
                        System.out.println("A helyes válasz: " + valasz[i - 1]);
                }

                sc1.close();
                sc2.close();

                System.out.println("\nA 7. feladat megoldása");
                System.out.println("Feladatsor generálása és kiíratása a tesztfel.txt fájlba");

                boolean[] kivalasztott = new boolean[db];

                for (i = 1; i <= db; i++) {
                        kivalasztott[i - 1] = false;
                }

                // Ismétlődés nélküli véletlen sorozat előállítása
                int[] feladatsor = new int[10];
                aktDb = 0;
                do {
                        sorszam = (int) (Math.random() * db) + 1;
                        if (!kivalasztott[sorszam - 1]) {
                                // még nem választottuk ki a feladatot
                                kivalasztott[sorszam - 1] = true;
                                // ez többet nem választható!
                                aktDb++;
                                feladatsor[aktDb - 1] = sorszam;
                        }
                } while (aktDb < 10);

                PrintWriter kiir = new PrintWriter(new FileWriter("tesztfel.txt"));

                int aktSorszam;
                int osszPont = 0;

                for (i = 1; i <= 10; i++) {
                        aktSorszam = feladatsor[i - 1];
                        osszPont += pont[aktSorszam - 1];
                        kiir.println(pont[aktSorszam - 1] + " " + valasz[aktSorszam - 1] + " " + kerdes[aktSorszam - 1]);
                }
                kiir.println("A feladatsorra összesen " + osszPont + " pont adható.");

                kiir.close();
                System.out.println("A tesztfel.txt fájl kiíratása megtörtént. \n");
        }

}
```

```
Az 1. feladat megoldása
A felszam.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
Az adatfájlban 49 feladat van.

A 3. feladat megoldása
A matematika feladatok jellemzése
Az adatfájlban 16 matematika feladat van,
1 pontot ér 5 feladat, 2 pontot ér 7 feladat, 3 pontot ér 4 feladat.

A 4. feladat megoldása
A válaszok számértékének terjedelme
A legkisebb: 2, a legnagyobb: 93030.

Az 5. feladat megoldása
Az adatfájlban szereplő témakörök:
tortenelem
foldrajz
magyar
kemia
matematika

A 6. feladat megoldása
Témakör bekérése - kérdés sorsolása - felelet bekérése - értékelés
Melyik témakörből szeretne kérdést kapni? Írja be ékezet nélkül: kemia
Mennyi az oxigen rendszama? 8
A válasz 2 pontot ér.

A 7. feladat megoldása
Feladatsor generálása és kiíratása a tesztfel.txt fájlba
A tesztfel.txt fájl kiíratása megtörtént.
```

A tesztfel.txt szövegfájl

```
3 9 Hany lakosa volt Magyarorszag legkisebb telepulesenek, Iborfianak, 2012 januar 1-en?
1 12 Mennyi 4 es 6 legkisebb kozos tobbszorose?
2 17 Hany orszag alkotja az eurozonat (2012-ben)
3 10 Hany orszagon halad at a Duna?
1 540 Hany fok a szabalyos otszog szogeinek osszege?
1 1222 Mikor adtak ki az Aranybullat?
1 1014 Milyen magas a Kekesteto
3 6370 Mennyi a Fold atlagos sugara (10 km-re kerekitve)?
2 24 Hany atloja van a szabalyos nyolcszognek?
2 1949 Melyik evben alakult meg a NATO?
A feladatsorra összesen 19 pont adható.
```

2013 október: Közúti ellenőrzés

Bizonyára mindenki látott már rendőrjárőrt, aki szolgálata során egy út menti ellenőrző pontról a forgalmat figyelte. A járőr feladata lehet a szabálytalankodók kiszűrése mellett az elhaladó járművek szúrópróbaszerű vagy módszeres ellenőrzése. Bizonyos esetekben egy műszaki ellenőrző állomás is kitelepül, amely alkalmas a kiválasztott járművek műszaki állapotának felmérésére.

Egy olyan nap adatait kell feldolgoznia, amelyen a rendőri mellett műszaki ellenőrzés is zajlott egy egyirányú út mentén. Az úton haladó legalább 50, de legfeljebb 1000 jármű adatait a *jarmu.txt* állományban tárolta el a rendőrautó forgalomrögzítő kamerájához csatlakoztatott gép. Az állomány sorai azonos szerkezetűek, az időt és a rendszámot tartalmazzák az elhaladás sorrendjében. A rendszám mindig 7 karakter hosszú, az angol ábécé nagybetűit, kötőjelet és számjegyeket tartalmaz ebben a sorrendben. A példában szereplőtől eltérő felépítésű rendszámok is lehetségesek.

Például:

```
11 12 05 TI-2342
11 12 09 BU-5523
11 12 41 AAAA-99
11 13 12 DM-5632
```

…

A 2. sor mutatja, hogy a BU-5523 jármű 11 óra 12 perc 9 másodperckor haladt át az ellenőrző ponton.

Készítsen programot, amely az alábbi kérdésekre válaszol! A program forráskódját mentse *jaror* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be a *jarmu.txt* állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat!
2. Határozza meg, hogy aznap legalább hány óra hosszat dolgoztak az ellenőrzést végzők, ha munkaidejük egész órakor kezdődik, és pontosan egész órakor végződik! (Minden óra 0 perc 0 másodperckor kezdődik, és 59 perc 59 másodperccel végződik.) Az eredményt jelenítse meg a képernyőn!
3. Műszaki ellenőrzésre minden órában egy járművet választanak ki. Azt, amelyik abban az órában először halad arra. Az ellenőrzés óráját és az ellenőrzött jármű rendszámát jelenítse meg a képernyőn a következő formában: `9 óra: AB-1234`! Minden óra adata külön sorba kerüljön! Csak azon órák adatai jelenjenek meg, amikor volt ellenőrizhető jármű!
4. A rendszám első karaktere külön jelentéssel bír. Az egyes betűk közül a „***B***” autóbuszt, a „***K***” kamiont, az „***M***” motort jelöl, a többi rendszámhoz személygépkocsi tartozik. Jelenítse meg a képernyőn, hogy az egyes kategóriákból hány jármű haladt el az ellenőrző pont előtt!
5. Mettől meddig tartott a leghosszabb forgalommentes időszak?
   A választ jelenítse meg a képernyőn a következő formában: `9:9:13 - 9:15:3`!

6. A rendőrök egy baleset közelében látott járművet keresnek rendszám alapján. A szemtanúk csak a rendszám bizonyos karaktereire emlékeztek, így a rendszám ismeretlen karaktereit a * karakterrel helyettesítve keresik a nyilvántartásban. Kérjen be a felhasználótól egy ilyen rendszámot, majd jelenítse meg a képernyőn az arra illeszthető rendszámokat!

7. Egy közúti ellenőrzés pontosan 5 percig tart. Amíg az ellenőrzés folyik, a járművek szabadon elhaladhatnak, a következő megállítására csak az ellenőrzés befejezése után kerül sor. Ha a rendőrök a legelső járművet ellenőrizték, akkor mely járműveket tudták ellenőrizni a szolgálat végéig? Írja az ellenőrzött járművek áthaladási idejét és rendszámát a *vizsgalt.txt* állományba az áthaladás sorrendjében, a bemenettel egyező formában! Ügyeljen arra, hogy az időadatokhoz tartozó számok a bevezető nullákat tartalmazzák!

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Közúti ellenőrzés
 *
 * @author Klemand
 */

public class EmeltInfo2013okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A jarmu.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("jarmu.txt"));

		int[] ora = new int[1000];
		int[] perc = new int[1000];
		int[] mp = new int[1000];
		String[] rendszam = new String[1000];
		int db = 0;
		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			daraboltSor = sor.split(" ");
			ora[db - 1] = Integer.parseInt(daraboltSor[0]);
			perc[db - 1] = Integer.parseInt(daraboltSor[1]);
			mp[db - 1] = Integer.parseInt(daraboltSor[2]);
			rendszam[db - 1] = daraboltSor[3];
		}

		beolvas.close();
		System.out.println("A beolvasás megtörtént. \n");

		System.out.println("A 2. feladat megoldása ");
		System.out.print("Az ellenőrzést végzők legalább ");
		System.out.println((ora[db - 1] - ora[0] + 1) + " órát dolgoztak. \n");

		System.out.println("A 3. feladat megoldása");
		System.out.println("Óránkénti ellenőrzés: ");
		int i;
		System.out.println(ora[0] + " óra: " + rendszam[0]);

		for (i = 2; i <= db; i++) {
			if (ora[i - 1] != ora[i - 2]) {
				System.out.println(ora[i - 1] + " óra: " + rendszam[i - 1]);
			}
		}
```

```
System.out.println("\nA 4. feladat megoldása ");
System.out.println("Kategóriánkénti statisztika az ellenőrzőpont előtti áthaladásról: ");

int[] katDb = new int[4];

for (i = 1; i <= 4; i++) {
        katDb[i - 1] = 0;
}

for (i = 1; i <= db; i++) {
        int kat = katMeghat(rendszam[i - 1]);
        katDb[kat - 1]++;
}

System.out.println("Autóbusz: " + katDb[0]);
System.out.println("Kamion: " + katDb[1]);
System.out.println("Motorkerékpár: " + katDb[2]);
System.out.println("Személygépkocsi: " + katDb[3]);

System.out.println("\nA 5. feladat megoldása");
System.out.print("A leghosszabb forgalommentes időszak: ");

int elozoIdo = ora[0] * 3600 + perc[0] * 60 + mp[0];
int aktIdo = ora[1] * 3600 + perc[1] * 60 + mp[1];
int max = 2;
int maxIdo = aktIdo - elozoIdo;

for (i = 3; i <= db; i++) {
        elozoIdo = aktIdo;
        aktIdo = ora[i - 1] * 3600 + perc[i - 1] * 60 + mp[i - 1];
        if (aktIdo - elozoIdo > maxIdo) {
                max = i;
                maxIdo = aktIdo - elozoIdo;
        }
}

System.out.print(ora[max - 2] + ":" + perc[max - 2] + ":" + mp[max - 2]);
System.out.println(" - " + ora[max - 1] + ":" + perc[max - 1] + ":" + mp[max - 1]);

System.out.println("\nA 6. feladat megoldása");
System.out.println("Egy hiányos rendszám keresése ");

Scanner sc = new Scanner(System.in);
System.out.print("Kérem a rendszámot (7 karakter), a hiányzó karaktereket *-gal jelölve: ");
String foszlany = sc.nextLine().toUpperCase();

System.out.println("A megfelelő rendszámok: ");
boolean megfelel;
int j;
int karDb = 7;
for (i = 1; i <= db; i++) {
        megfelel = true;
        j = 1;
        while (j <= karDb && megfelel) {
                if (!foszlany.substring(j - 1, j).equals("*")) {
                        if (!rendszam[i - 1].substring(j - 1, j).equals(foszlany.substring(j - 1, j))) {
                                megfelel = false;
                        }
                }
                j++;
        }

        if (megfelel) {
                System.out.println(rendszam[i - 1]);
        }
}
sc.close();
```

```
                System.out.println("\nA 7. feladat megoldása");
                System.out.println("Az ellenőrzött járművek kiíratása a vizsgalt.txt fájlba");

                PrintWriter kiir = new PrintWriter(new FileWriter("vizsgalt.txt"));

                int elozoEll = ora[0] * 3600 + perc[0] * 60 + mp[0];

                kiir.println(idoKonvertalas(elozoEll) + " " + rendszam[0]);

                for (i = 2; i <= db; i++) {
                        aktIdo = ora[i - 1] * 3600 + perc[i - 1] * 60 + mp[i - 1];
                        if (aktIdo - elozoEll >= 300) {
                                kiir.println(idoKonvertalas(aktIdo) + " " + rendszam[i - 1]);
                                elozoEll = aktIdo;
                        }
                }
                kiir.close();
                System.out.println("A vizsgalt.txt fájl kiíratása megtörtént. \n");

        }

        public static int katMeghat(String rendszam) {
                String elso = rendszam.substring(0, 1);
                switch (elso) {
                case "B":
                        return 1;
                case "K":
                        return 2;
                case "M":
                        return 3;
                default:
                        return 4;
                }
        }

        public static String idoKonvertalas(int ido) {
                int ora, perc, mp;
                String konv = "";
                mp = ido % 60;
                perc = ido / 60;
                ora = perc / 60;
                perc = perc % 60;

                if (ora < 10) {
                        konv += "0";
                }
                konv += ora + " ";

                if (perc < 10) {
                        konv += "0";
                }
                konv += perc + " ";

                if (mp < 10) {
                        konv += "0";
                }
                konv += mp;

                return konv; // visszaadott függvényérték
        }

}
```

```
Az 1. feladat megoldása
A jarmu.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
Az ellenőrzést végzők legalább 6 órát dolgoztak.

A 3. feladat megoldása
Óránkénti ellenőrzés:
8 óra: FD-2717
9 óra: GK-3407
10 óra: RQ-8890
11 óra: IN-5066
12 óra: GC-0459
13 óra: CH-1893

A 4. feladat megoldása
Kategóriánkénti statisztika az ellenőrzőpont előtti áthaladásról:
Autóbusz: 10
Kamion: 12
Motorkerékpár: 15
Személygépkocsi: 317

A 5. feladat megoldása
A leghosszabb forgalommentes időszak: 8:57:48 - 9:1:6

A 6. feladat megoldása
Egy hiányos rendszám keresése
Kérem a rendszámot (7 karakter), a hiányzó karaktereket *-gal jelölve: *****56
A megfelelő rendszámok:
SM-6556
LX-6656
UW-6256
QA-4856
CL-1656

A 7. feladat megoldása
Az ellenőrzött járművek kiíratása a vizsgalt.txt fájlba
A vizsgalt.txt fájl kiíratása megtörtént.
```

A vizsgalt.txt szövegfájl

```
08 46 51 FD-2717
08 51 51 FY-2063
08 57 07 LT-4076
09 02 21 EB-2944
09 07 55 VS-1521
09 13 09 RW-5733
09 19 14 SM-6556
09 24 14 XP-4672
09 29 14 YA-7536
09 34 35 FU-2341
09 40 24 XL-7193
09 45 59 II-4392
09 51 04 CK-0726
09 56 31 VD-9088
10 01 50 MJ-4313
10 07 02 LN-5680
10 12 04 NR-1269
10 17 20 CG-4491
10 22 24 YO-0524
10 28 19 BB-8519
10 34 29 XG-0235
10 39 57 DJ-9459
10 45 03 DH-2465
10 50 52 YZ-0459
10 56 31 VB-9322
11 02 53 IX-2030
11 08 23 VQ-9592
11 13 49 HW-1538
11 18 55 VG-2275
11 23 59 LJ-7985
11 29 46 PY-6651
11 35 34 KC-5813
11 41 18 IV-3641
11 46 33 KQ-2692
11 51 50 FF-4282
11 57 34 CH-5530
12 03 14 PH-8255
12 09 26 XD-7831
12 15 01 AC-9946
12 20 04 EL-6483
12 25 17 RS-2750
12 30 41 HU-6881
12 36 05 ZG-4536
12 41 18 OQ-8017
12 46 27 AS-8521
12 52 22 UB-9408
12 57 40 JM-4042
13 03 00 EM-3026
13 08 27 GL-2740
13 14 08 PQ-8950
13 19 23 JJ-0608
13 24 59 PM-2524
13 30 05 SJ-5336
13 35 10 IS-3397
13 40 44 RK-3908
13 46 35 BD-5782
```

2014 május: IPv6

A számítógépes hálózatok üzemeltetésében az IPv4-es címeket lassan leváltja az IPv6-os címzési rendszer, amely az eddigi 32 bit hosszúságú címek helyett 128 bit hosszúságú címeket használ.

Az IPv6-os címeket hexadecimális alakban ábrázoljuk, nyolc darab négyes csoportba osztva. Az egyes számjegyek a tízes számrendszerben is használt számjegyek, valamint az *a*, *b*, *c*, *d*, *e*, *f* betűk lehetnek. Az egyes csoportokat kettősponttal választjuk el. Ezek alapján formailag megfelelő IPv6-os cím a következő:

`2001:0db8:03cd:0000:0000:ef45:0006:0123`

Egy nagyvállalatnál készítettek egy programot, ami a cég szerverén tárolt összes dokumentumból kigyűjtötte az IPv6-címeket. Az így keletkezett gyűjteményt az *ip.txt* fájl tárolja. Minden IP-címet csak az első előfordulásakor rögzítettek. Az állomány legalább 20, de legfeljebb 500 adatsort, soronként egy IP-címet tartalmaz a következő példának megfelelően:

```
2001:0db8:03cd:0000:0000:ef45:0006:0123
2001:0e10:0000:aabc:0000:01ac:0000:0001
fdf8:f53b:82e4:0000:0000:0000:0000:0053
fc00:0000:0000:ad65:0124:33ab:0100:6543 …
```

A vállalatnál háromféle IP-cím fordul elő. A feladat megoldásában csak ezekkel a címekkel kell foglalkozni:

- A `2001:0db8` kezdetű címek a *dokumentációs cím*ek, eszközöknek nincsenek kiosztva.
- A `2001:0e` kezdetű címek az informatikai eszközöknek kiosztott *globális egyedi cím*ek.
- Az `fc`, valamint az `fd` kezdetű címek az eszközöknek kiosztott *helyi egyedi cím*ek.

Több szabály vonatkozik a címek rövidebb leírásának lehetőségére:

- Az egyes csoportokban a bevezető nullák elhagyhatók. Például így leírva a fenti cím: `2001:db8:3cd:0:0:ef45:6:123`
- Kettő vagy több csak nullákból álló csoportot le lehet egyszerűsíteni két kettőspont közötti üres csoportra. Ezzel a szabállyal tovább egyszerűsítve az előző címet: `2001:db8:3cd::ef45:6:123`
- Ha egy címben több helyen is vannak csak nullákból álló csoportok, akkor is csak egyszer lehet ez utóbbi módszerrel rövidítést végrehajtani. Ilyen esetben mindig a több nullás csoportot kell rövidíteni. Ha azonos számú nullás csoport található a címen belül több helyen is, akkor balról az elsőt kell rövidíteni.

  Például: `2001:0000:0000:00f5:0000:0000:0000:0123`

  Rövidítve: `2001:0:0:f5::123`

Készítsen programot, amely az *ip.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *cimek* néven! (A program megírásakor a megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a

képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott. A képernyőre írást igénylő feladatok eredményét a feladatok utáni mintának megfelelően jelenítse meg!

1. Olvassa be az *ip.txt* állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat!
2. Határozza meg és írja a képernyőre, hogy hány adatsor van az állományban!
3. Írja a képernyőre az állományban található legalacsonyabb IP-címet! A megoldásában felhasználhatja, hogy a betűk ASCII-kódjai a számok ASCII-kódjai után találhatók a kódtáblában.
4. Határozza meg, hogy az állományban hány darab IP-cím van az egyes fajtákból! Az eredményt jelenítse meg a képernyőn a mintának megfelelően!
5. Gyűjtse ki a *sok.txt* állományba azokat az IP-címeket, melyek legalább 18 nullát tartalmaznak! A fájlban minden sor elején szerepeljen az eredeti állományból a cím sorszáma! Ezt kövesse egy szóközzel elválasztva a cím az *ip.txt* állományban szereplő alakjával!
6. Kérjen be a felhasználótól egy sorszámot! Az állományban a megadott sorszámon található IP-címet rövidítse a csoportokon belüli bevezető nullák elhagyásával! Az állományban található alakot és a rövidített változatot írja a képernyőre egymás alá!
7. Az előző feladatban használt IP-címet rövidítse tovább az egymást követő nullás csoportok rövidítésére vonatkozó szabályoknak megfelelően! Az eredményt jelenítse meg a képernyőn! Amennyiben nem rövidíthető, írja ki: „`Nem rövidíthető tovább.`”!

   Minta a szöveges kimenetek kialakításához:

```
2. feladat:
Az állományban 372 darab adatsor van.

3. feladat:
A legalacsonyabb tárolt IP-cím:
2001:0db8:0000:00b9:0800:0f00:e02a:71ac

4. feladat:
Dokumentációs cím: 106 darab
Globális egyedi cím: 120 darab
Helyi egyedi cím: 146 darab

6. feladat:
Kérek egy sorszámot: 10
fcef:b0e7:7d20:0000:0000:0000:3b95:0565
fcef:b0e7:7d20:0:0:0:3b95:565

7. feladat: fcef:b0e7:7d20::3b95:565
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * IPv6
 *
 * @author Klemand
 */

public class EmeltInfo2014maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása az ip.txt fájlból");

		BufferedReader beolvas = new BufferedReader(new FileReader("ip.txt"));

		String[] ipCim = new String[500];
		int db = 0;
		String sor;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			ipCim[db - 1] = sor;
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Az IPv6 címek számának meghatározása");

		System.out.println("Az állományban " + db + " darab IPv6 cím van. \n");

		System.out.println("A 3. feladat megoldása");
		System.out.println("A legalacsonyabb tárolt IP-cím: ");

		int i;
		int min = 1;
		for (i = 2; i <= db; i++) {
			if (ipCim[i - 1].compareTo(ipCim[min - 1]) < 0) {
				min = i;
			}
		}
		System.out.println(ipCim[min - 1]);
```

```
        System.out.println("\nA 4. feladat megoldása");
        System.out.println("Az IPv6 címek fajtái");

        int dokDb = 0;
        int globDb = 0;
        int helyiDb = 0;

        for (i = 1; i <= db; i++) {
            if (ipCim[i - 1].substring(0, 9).equals("2001:0db8")) {
                dokDb++;
            } else if (ipCim[i - 1].substring(0, 7).equals("2001:0e")) {
                globDb++;
            } else {
                helyiDb++;
            }
        }
        System.out.println("Dokumentációs cím: " + dokDb + " darab");
        System.out.println("Globális cím: " + globDb + " darab");
        System.out.println("Helyi egyedi cím: " + helyiDb + " darab");

        System.out.println("\nAz 5. feladat megoldása");
        System.out.println("A legalább 18 db nullát tartalmazó IP-címek kiírása a sok.txt fájlba ");

        PrintWriter kiir = new PrintWriter(new FileWriter("sok.txt"));

        for (i = 1; i <= db; i++) {
            if (sokE(ipCim[i - 1])) {
                kiir.println(i + " " + ipCim[i - 1]);
            }
        }
        kiir.close();

        System.out.println("A fájlkiírás megtörtént.\n");

        System.out.println("A 6. feladat megoldása");
        System.out.println("Egy bekért sorszámú IP-cím rövidítése");

        Scanner sc = new Scanner(System.in);
        System.out.print("Kérek egy IP-sorszámot (max " + db + "): ");
        int sorszam = sc.nextInt();

        System.out.println(ipCim[sorszam - 1]);
        String ipRovid = ipRovidites(ipCim[sorszam - 1]);
        System.out.println(ipRovid);
        sc.close();

        System.out.println("\nA 7. feladat megoldása");
        System.out.println("Az előbbi IP-cím további rövidítése");
        String ipUltraRovid = ipUltraRovidites(ipRovid);
        System.out.println(ipUltraRovid);
        System.out.println("");

    }

    public static boolean sokE(String cim) {
        int nullDb = 0;

        for (int j = 1; j <= cim.length(); j++) {
            if (cim.substring(j - 1, j).equals("0")) {
                nullDb++;
            }
        }
        return (nullDb >= 18);
    }
```

```
	public static String ipRovidites(String ip) {
		String[] cim = ip.split(":");
		String ipR = "";
		String aktCim;
		int j;
		for (int i = 1; i <= 8; i++) {
			aktCim = cim[i - 1];
			j = 1;
			while (j <= 3 && aktCim.substring(0, 1).equals("0")) {
				aktCim = aktCim.substring(1);
				j++;
			}
			ipR += aktCim;
			if (i < 8) {
				ipR += ":";
			}
		}
		return ipR;
	}

	public static String ipUltraRovidites(String ip) {
		String[] cim = ip.split(":");
		String ipUR = "";

		int aktTkezdet = -1; // irreális kezdőérték
		int aktThossz = 0;
		int maxTkezdet = -1; // irreális kezdőérték
		int maxThossz = 0;

		for (int i = 1; i <= 8; i++) {
			if (cim[i - 1].equals("0")) { // ez a T tulajdonság
				if (aktTkezdet == -1) {
					aktTkezdet = i;
				}
				aktThossz++;
				if (aktThossz > maxThossz) {
					maxTkezdet = aktTkezdet;
					maxThossz = aktThossz;
				}
			} else {
				aktTkezdet = -1;
				aktThossz = 0;
			}
		}

		if (maxThossz >= 2) {
			for (int i = 1; i < maxTkezdet; i++) {
				ipUR += cim[i - 1] + ":";
			}
			ipUR += ":";

			for (int i = maxTkezdet + maxThossz; i <= 8; i++) {
				ipUR += cim[i - 1];
				if (i < 8) {
					ipUR += ":";
				}
			}
		} else {
			ipUR = "Nem rövidíthető tovább.";
		}
		return ipUR;
	}

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása az ip.txt fájlból
A beolvasás megtörtént.

A 2. feladat megoldása
Az IPv6 címek számának meghatározása
Az állományban 375 darab IPv6 cím van.

A 3. feladat megoldása
A legalacsonyabb tárolt IP-cím:
2001:0db8:0000:00b9:0800:0f00:e02a:71ac

A 4. feladat megoldása
Az IPv6 címek fajtái
Dokumentációs cím: 106 darab
Globális cím: 120 darab
Helyi egyedi cím: 149 darab

Az 5. feladat megoldása
A legalább 18 db nullát tartalmazó IP-címek kiírása a sok.txt fájlba
A fájlkiírás megtörtént.

A 6. feladat megoldása
Egy bekért sorszámú IP-cím rövidítése
Kérek egy IP-sorszámot (max 375): 373
fc11:0000:0000:0f00:0000:0000:0000:2222
fc11:0:0:f00:0:0:0:2222

A 7. feladat megoldása
Az előbbi IP-cím további rövidítése
fc11:0:0:f00::2222


A sok.txt szövegfájl

1 fc00:0610:0f00:89f0:00f0:0ed2:0000:000d
6 2001:0db8:0030:1a90:0200:9000:c000:0088
8 fc0c:2200:00d0:0db0:0900:0a05:0000:00ef
11 fc10:f000:0700:08d0:c000:0000:0000:a08b
16 2001:0e04:0600:4e00:903d:a001:0000:000b
26 fc0d:3505:9000:0fc0:06c0:0030:0000:0033
30 fc0e:00e0:0002:0000:009d:0400:1000:0097
35 2001:0e10:0290:00b0:010d:20b0:0008:00bf
40 2001:0e00:0800:2000:0000:3009:0000:730e
47 2001:0e00:b010:0f00:5500:10df:e00b:0009
50 fc00:0e00:00b1:100c:7420:0007:0064:0008
54 2001:0e00:0000:0040:03a7:0000:0009:800b
61 2001:0e9d:e000:0000:0400:0400:00fc:0c0c
62 2001:0e00:0000:ab00:0000:141e:3f0f:0c05
64 fd00:0000:c000:b008:1600:0c0f:d600:0b04
68 2001:0db8:0b20:1000:0880:000d:0000:a06a
69 fc00:8c00:00b0:0030:00c0:0e00:02c9:f00a
82 2001:0e00:0500:50d0:0830:9005:9400:d00f
89 2001:0e90:0050:0400:00d0:0d00:0150:8f62
92 fc20:b10e:0000:0200:0107:4f00:100c:0005
94 2001:0db8:0066:004f:0000:0c60:000a:0f06
96 2001:0e00:dd00:9400:0004:0000:00e8:0823
97 fc00:05c0:3480:00a0:0000:0000:0bc0:e01e
105 2001:0e40:d000:090f:8000:2f00:000b:009b
106 fc10:0504:20be:0060:0000:000e:500e:00ba
110 fd20:0000:1001:0006:300b:0401:0c88:0003
123 2001:0e00:100b:0400:0b00:64ab:0200:0073
128 2001:0e00:006e:0638:000f:0000:400c:0f0e
140 2001:0db8:a700:2a00:0000:0dda:0200:0003
...
```

2014 május idegennyelvű: Céllövészet

A Sor Lövészegylet rendszeresen rendez versenyt az alábbi, igen egyszerű szabályokkal:

- A lövések leadására korlátozott idő áll rendelkezésre, ezért a versenyzők eltérő számú lövést adhatnak le.
- A lövéseket sorszámozott korongokra kell leadni.
- Találatnak számít, ha a korongot bárhol érinti a lövedék.
- A lövésekhez pontértéket rendelnek: amíg nem hibázik valaki, minden találata 20 pontot ér; de rontás esetén minden hiba 1 ponttal csökkenti – egészen nulláig – a későbbi lövésekkel szerezhető pontszámot. A lövés pontértéke nem lehet negatív.
- Az végez előrébb a versenyben, aki több pontot szerez. A holtversenyt nem döntik el, mindegyik versenyző ugyanolyan helyezéssel végez, tehát mindenki helyezése megegyezik a nála több pontot szerzett versenyzők számánál eggyel nagyobb számmal.

A *verseny.txt* állományban versenyzőnként feljegyeztük a lövések eredményét. A fájl első sorában a versenyzők száma ($2 \leq v \leq 100$) szerepel. A következő v sorban legfeljebb l ($4 \leq l \leq 40$) karakter található, egy versenyző lövéseinek sorozata. Egy lövést egy karakter ír le, a – karakter a sikertelen, a + karakter a sikeres lövést rögzíti.

Például:

```
5
+--+
-+-+++-
-+--+--
++---
-++--
```

A példában a 4. sor azt mutatja, hogy a 3-as rajtszámú lövőnek a 2. és az 5. lövése talált, tehát a versenyző csak két korongot talált el. Mivel elsőre hibázott, az első találat 19 pontot ér, aztán a két újabb hiba miatt már csak 17 pontot jelentett a második találat. Tehát összesen 36 pontot szerzett. Az 5. sorban szereplő, 4-es rajtszámú versenyző ugyancsak 2 találattal 40 pontot szerzett.

Készítsen programot, amely a *verseny.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *loves* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`), az 5. feladat esetén pedig a részfeladat betűjelét is! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be a *verseny.txt* állományban található adatokat, és annak felhasználásával oldja meg a következő feladatokat!
2. Írja a képernyőre azon versenyzők rajtszámát, akiknek egymás után két (vagy több) lövése is talált! A versenyzők rajtszámát egy-egy szóközzel válassza el egymástól!
3. Írja a képernyőre, hogy melyik versenyző adta le a legtöbb lövést! Ha többen is ugyanannyi lövést adtak le, elegendő egyikük rajtszámát kiírni.

4. Készítsen függvényt *loertek* néven az alábbi algoritmus alapján! A függvény egy + és – jeleket tartalmazó, legfeljebb 40 hosszúságú karaktersorozathoz hozzárendeli a feladatban képviselt pontértékét. A függvény elkészítésekor az algoritmusban megadott változóneveket használja! Az elkészített függvényt a további feladatok megoldásánál használja fel! A függvény bemenő paramétere az egy játékos lövéseit leíró karaktersorozat, értéke pedig az ahhoz rendelt pontszám.

```
Függvény   loertek(sor:karaktersorozat):egész   szám
     aktpont:=20
     ertek:=0
     Ciklus i:=1-től hossz(sor)-ig
         Ha aktpont>0 és sor[i]="-" akkor
            aktpont:=aktpont-1
         Különben
            ertek:=ertek+aktpont
         Elágazás vége
     Ciklus vége
     loertek:=ertek
Függvény vége
```

5. Kérje be a felhasználótól egy versenyző sorszámát, majd írja ki, hogy:

   a. hányadik lövései találtak (az értékeket egymástól szóközzel válassza el!)

   b. hány korongot talált el összesen

   c. milyen hosszú volt a leghosszabb hibátlan lövéssorozata

   d. hány pontot ért el!

   Az eredmény megjelenítése előtt írja képernyőre a részfeladat betűjelét is!

6. Állítsa elő a *sorrend.txt* állományban a verseny végeredményét! A fájlban soronként tüntesse fel a versenyző helyezését, rajtszámát és pontszámát! A helyezés megadásakor a holtversenyt a bevezetőben megfogalmazott szabályok alapján az alábbi mintához hasonlóan kezelje! Az adatokat egy-egy tabulátorral (ASCII kódja a 9-es) válassza el egymástól! A lista legyen pontszám szerint csökkenő!

   Például a feladat elején olvasható példa bemenet esetén a fájl tartalma:

```
1    2    73
2    4    40
3    1    38
3    5    38
5    3    36
```

Példa a szöveges kimenetek kialakításához:

```
2. feladat:
Az egymast kovetoen tobbszor talalo versenyzok: 2 4 5
3. feladat:
A legtobb lovest leado versenyzo rajtszama: 2
5. feladat:
Adjon meg egy rajtszamot! 2
5a. feladat: Celt ero lovesek: 2 4 5 6
5b. feladat: Az eltalalt korongok szama: 4
5c. feladat: A leghosszabb hibatlan sorozat hossza: 3
5d. feladat: A versenyzo pontszama: 73
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Céllövészet
 *
 * @author Klemand
 */

public class EmeltInfo2014mid {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az adatok beolvasása a verseny.txt fájlból");

		BufferedReader beolvas = new BufferedReader(new FileReader("verseny.txt"));

		String sor;
		sor = beolvas.readLine().trim();
		int db = Integer.parseInt(sor);
		String[] lovesek = new String[db];

		int i;
		for (i = 1; i <= db; i++) {
			lovesek[i - 1] = beolvas.readLine();
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Az egymást követően többször találó versenyzők:");

		for (i = 1; i <= db; i++) {
			if (lovesek[i - 1].contains("++")) {
				System.out.print(i + " ");
			}
		}

		System.out.println("\n\nA 3. feladat megoldása");
		System.out.print("A legtöbb lövést leadó versenyző rajtszáma: ");

		int max = 1;
		for (i = 2; i <= db; i++) {
			if (lovesek[i - 1].length() > lovesek[max - 1].length()) {
				max = i;
			}
		}
		System.out.println(max + "\n");

		System.out.println("A 4. feladat megoldása");
		System.out.println("A loertek függvény elkészítése\n");
		// A függvény a main metódus után található
```

```
System.out.println("Az 5. feladat megoldása");
System.out.println("Egy versenyző sorszámának bekérése és eredményének kiértékelése");

Scanner sc = new Scanner(System.in);
System.out.print("Kérem a versenyző sorszámát (max " + db + "): ");

int rajtszam = sc.nextInt();
String sorozat = lovesek[rajtszam - 1];
int hossz = sorozat.length();

System.out.println("5. a.");
System.out.print("Célt érő lövések:");

for (i = 1; i <= hossz; i++) {
        if (sorozat.substring(i - 1, i).equals("+")) {
                System.out.print(" " + i);
        }
}

System.out.println("\n5. b.");
System.out.print("Az eltalált korongok száma: ");

int talalatDb = 0;
for (i = 1; i <= hossz; i++) {
        if (sorozat.substring(i - 1, i).equals("+")) {
                talalatDb++;
        }
}
System.out.println(talalatDb);

System.out.println("5. c.");
System.out.print("A leghosszabb hibátlan lövéssorozatának hossza: ");
// Az most nem kell, hogy hol kezdődik a hibátlan lövéssorozat.

int maxThossz = 0;
int aktThossz = 0;

for (i = 1; i <= hossz; i++) {
        if (sorozat.substring(i - 1, i).equals("+")) {
                // Ez a T tulajdonság
                aktThossz++;
                if (aktThossz > maxThossz) {
                        maxThossz = aktThossz;
                }
        } else {
                aktThossz = 0;
        }
}
System.out.println(maxThossz);

System.out.println("5. d.");
System.out.print("A lőeredménye: ");

int pontszam = loertek(sorozat);
System.out.println(pontszam + " pont \n");
sc.close();
```

```
		System.out.println("A 6. feladat megoldása");
		System.out.println("A verseny végeredményének kiíratása a sorrend.txt fájlba");

		int[] rajtszamok = new int[db];
		int[] pontszamok = new int[db];
		for (i = 1; i <= db; i++) {
			rajtszamok[i - 1] = i;
			pontszamok[i - 1] = loertek(lovesek[i - 1]);
		}

		// A rajtszámok rendezése pontszámok szerint csökkenő sorrendben
		int j, asztal;
		for (i = db - 1; i >= 1; i--) {
			for (j = 1; j <= i; j++) {
				if (pontszamok[j - 1] < pontszamok[j]) {
					asztal = rajtszamok[j - 1];
					rajtszamok[j - 1] = rajtszamok[j];
					rajtszamok[j] = asztal;

					asztal = pontszamok[j - 1];
					pontszamok[j - 1] = pontszamok[j];
					pontszamok[j] = asztal;
				}
			}
		}

		PrintWriter kiir = new PrintWriter(new FileWriter("sorrend.txt"));

		int aktHelyezes = -1;
		int aktPont = 1000000; // irreálisan magas kezdőérték
		for (i = 1; i <= db; i++) {
			if (pontszamok[i - 1] < aktPont) {
				aktHelyezes = i;
				aktPont = pontszamok[i - 1];
			}

			kiir.print(aktHelyezes + "\t" + rajtszamok[i - 1]);
			kiir.println("\t" + aktPont);
		}
		kiir.close();
		System.out.println("A sorrend.txt fájl kiíratása befejeződött. \n");

	}

	public static int loertek(String sor) {
		int aktPont = 20;
		int ertek = 0;
		for (int i = 1; i <= sor.length(); i++) {
			if (aktPont > 0 && sor.substring(i - 1, i).equals("-")) {
				aktPont -= 1;
			} else {
				ertek += aktPont;
			}
		}
		int loertek = ertek;
		return loertek;
	}

}
```

```
Az 1. feladat megoldása
Az adatok beolvasása a verseny.txt fájlból
A beolvasás megtörtént.

A 2. feladat megoldása
Az egymást követően többször találó versenyzők:
2 4 5 6 7 8 10 12 14 15 16 18 19 20 21

A 3. feladat megoldása
A legtöbb lövést leadó versenyző rajtszáma: 20

A 4. feladat megoldása
A loertek függvény elkészítése

Az 5. feladat megoldása
Egy versenyző sorszámának bekérése és eredményének kiértékelése
Kérem a versenyző sorszámát (max 21): 8
5. a.
Célt érő lövések: 2 4 5 6 8
5. b.
Az eltalált korongok száma: 5
5. c.
A leghosszabb hibátlan lövéssorozatának hossza: 3
5. d.
A lőeredménye: 90 pont

A 6. feladat megoldása
A verseny végeredményének kiíratása a sorrend.txt fájlba
A sorrend.txt fájl kiíratása befejeződött.
```

A verseny.txt forrásfájl

```
21
+--+
--+++-
-+--+--
++---
-++--
--+-+++-----------------+-+
+-++--+
-+-++++-+
-+---+-+
--++++-
-+--+-
++--
-+--
-++-----------------+-+
+-++-+--+
-+-+++-
-+--+-+-
++---+
-++-+-
--+-+++---+-------------+-+
+-++--+
```

A sorrend.txt szövegfájl

1	15	92
2	8	90
3	20	84
4	7	75
4	21	75
6	16	73
7	6	70
8	18	57
9	19	56
10	2	54
10	10	54
12	9	52
12	17	52
14	14	43
15	4	40
15	12	40
17	1	38
17	5	38
19	3	36
19	11	36
21	13	19

2014 október: Nézőtér

A Fregoli Színházban a jegyeladásokat elektronikusan rögzítik. A színházban 15 sor, és soronként 20 szék van. A sorokat 1-től 15-ig számozzák, a sorokon belül pedig a székeket 1-től 20-ig. Egy előadásra a pillanatnyilag eladott jegyek eloszlását a *foglaltsag.txt* szöveges állomány tartalmazza, melyben „x” jelzi a foglalt és „o” a szabad helyeket.

Például:

```
ooxxxoxoxoxoxxxooxxx
xxxxxxxxxxxxxxxxxxxx
oxxxxxoooxxxxxxxxxxo
…
```

Az első sor 1-2. széke például még szabad, míg a 2. sorba az összes jegyet eladták.

A jegyek ára nem egyforma, összege a helytől függően ötféle lehet. Azt, hogy az adott szék az öt közül melyik árkategóriába tartozik, a *kategoria.txt* fájl tartalmazza az alábbi formában:

Például:

```
33222111111111122233
43322221111112222334
44433322222222333444
…
```

A példa szerint az 1. sor 2. széke a 3. kategóriába, a 2. sor 1. széke a 4. kategóriába esik.

Készítsen programot *nezoter* néven a következő feladatok megoldására! A program futása során a képernyőre való kiíráskor, illetve az adatok billentyűzetről való beolvasásakor utaljon a feladat sorszámára (például: `3. feladat`), és a kiírandó, illetve bekérendő tartalomra! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el a *foglaltsag.txt* és a *kategoria.txt* fájl adatait!
2. Kérje be a felhasználótól egy sor, és azon belül egy szék számát, majd írassa ki a képernyőre, hogy az adott hely még szabad-e vagy már foglalt!
3. Határozza meg, hogy hány jegyet adtak el eddig, és ez a nézőtér befogadóképességének hány százaléka! A százalékértéket kerekítse egészre, és az eredményt a következő formában írassa ki a képernyőre:

   Például: „`Az előadásra eddig 156 jegyet adtak el, ez a nézőtér 42%-a.`”
4. Határozza meg, hogy melyik árkategóriában adták el a legtöbb jegyet!
   Az eredményt írassa ki a képernyőre az alábbi formában:

   Például: „`A legtöbb jegyet a(z) 3. árkategóriában értékesítették.`”
5. A jegyek árát kategóriánként a következő táblázat tartalmazza:

árkategória	1	2	3	4	5
ár (Ft)	5000	4000	3000	2000	1500

   Mennyi lenne a színház bevétele a pillanatnyilag eladott jegyek alapján?
   Írassa ki az eredményt a képernyőre!

6. Mivel az emberek általában nem egyedül mennek színházba, ha egy üres hely mellett nincs egy másik üres hely is, akkor azt nehezebben lehet értékesíteni. Határozza meg, és írassa ki a képernyőre, hogy hány ilyen „egyedülálló” üres hely van a nézőtéren!

7. A színház elektronikus eladási rendszere az érdeklődőknek az üres helyek esetén a hely árkategóriáját jeleníti meg, míg a foglalt helyeket csak egy „x” karakterrel jelzi.
Készítse el ennek megfelelően a fenti adatokat tartalmazó *szabad.txt* fájlt!

Például:

```
33xxx1x1x1x1xxx22xxx
xxxxxxxxxxxxxxxxxxxx
4xxxxx222xxxxxxxxxx4
…
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Nézőtér
 *
 * @author Klemand
 */

public class EmeltInfo2014okt {

	public static void main(String[] args) throws IOException {

		System.out.println("A 1. feladat megoldása");
		System.out.println("A foglaltsag.txt és a kategoria.txt fájlok beolvasása ");

		BufferedReader beolvas = new BufferedReader(new FileReader("foglaltsag.txt"));

		int sorDb = 15;
		int helyDb = 20;
		String[][] foglaltsag = new String[sorDb][helyDb];
		String[] daraboltSor;

		int i, j;
		for (i = 1; i <= sorDb; i++) {
			daraboltSor = beolvas.readLine().split("");
			for (j = 1; j <= helyDb; j++) {
				foglaltsag[i - 1][j - 1] = daraboltSor[j - 1];
			}
		}
		beolvas.close();
		beolvas = new BufferedReader(new FileReader("kategoria.txt"));

		int[][] kategoria = new int[sorDb][helyDb];

		for (i = 1; i <= sorDb; i++) {
			daraboltSor = beolvas.readLine().split("");
			for (j = 1; j <= helyDb; j++) {
				kategoria[i - 1][j - 1] = Integer.parseInt(daraboltSor[j - 1]);
			}
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Egy bekért hely foglaltságának eldöntése ");

		Scanner sc = new Scanner(System.in);
		System.out.print("Kérem a sor számát (max 15): ");
		int sor = sc.nextInt();
		System.out.print("Kérem a szék számát (max 20): ");
		int szek = sc.nextInt();

		if (foglaltsag[sor - 1][szek - 1].equals("x")) {
			System.out.println("A kiválasztott hely már foglalt.");
		} else {
			System.out.println("A kiválasztott hely még szabad.");
		}
		sc.close();
```

```
System.out.println("\nA 3. feladat megoldása");
System.out.println("Hány jegyet adtak el eddig?");

int nezoterDb = 300;
int eladottDb = 0;

for (i = 1; i <= sorDb; i++) {
        for (j = 1; j <= helyDb; j++) {
                if (foglaltsag[i - 1][j - 1].equals("x")) {
                        eladottDb++;
                }
        }
}
System.out.print("Az előadásra eddig " + eladottDb + " darab jegyet adtak el, ez a nézőtér ");
double szazalek = ((double) eladottDb / nezoterDb) * 100;
System.out.printf("%.0f", szazalek); //Nem levág, hanem kerekít!
System.out.println("%-a. \n");

System.out.println("A 4. feladat megoldása");
System.out.println("Melyik árkategóriából adták el a legtöbb jegyet?");

int katDb = 5;
int[] katEladottDb = new int[katDb];

for (i = 1; i <= katDb; i++) {
        katEladottDb[i - 1] = 0; // kezdőértékek
}

int aktKat;
for (i = 1; i <= sorDb; i++) {
        for (j = 1; j <= helyDb; j++) {
                if (foglaltsag[i - 1][j - 1].equals("x")) {
                        aktKat = kategoria[i - 1][j - 1];
                        katEladottDb[aktKat - 1]++;
                }
        }
}

int max = 1;
for (i = 2; i <= katDb; i++) {
        if (katEladottDb[i - 1] > katEladottDb[max - 1]) {
                max = i;
        }
}
System.out.println("A legtöbb jegyet a(z) " + max + ". árkategóriában értékesítették. \n");

System.out.println("Az 5. feladat megoldása ");
System.out.println("Az aktuális bevétel meghatározása");

int bevetel = 0;
for (i = 1; i <= sorDb; i++) {
        for (j = 1; j <= helyDb; j++) {
                if (foglaltsag[i - 1][j - 1].equals("x")) {
                        switch (kategoria[i - 1][j - 1]) {
                        case 1:
                                bevetel += 5000;
                                break;
                        case 2:
                                bevetel += 4000;
                                break;
                        case 3:
                                bevetel += 3000;
                                break;
                        case 4:
                                bevetel += 2000;
                                break;
                        default:
                                bevetel += 1500;
                        }
                }
        }
}
System.out.println("A színház jelenlegi bevétele " + bevetel + " Ft lenne. \n");
```

```
                System.out.println("A 6. feladat megoldása");
                System.out.println("Egyedülálló üres helyek vizsgálata");

                int egyedulalloDb = 0;
                String aktHelyek;

                for (i = 1; i <= sorDb; i++) {
                        aktHelyek = "x" + foglaltsag[i - 1][0];
                        for (j = 2; j <= helyDb; j++) {
                                aktHelyek += foglaltsag[i - 1][j - 1];
                                if (aktHelyek.equals("xox")) {
                                        egyedulalloDb++;
                                }
                                aktHelyek = aktHelyek.substring(1);
                        }
                        if (aktHelyek.equals("xo")) {
                                egyedulalloDb++;
                        }
                }
                System.out.println("A nézőtéren " + egyedulalloDb + " db egyedülálló üres hely van.\n");


                System.out.println("A 7. feladat megoldása");
                System.out.println("A szabad helyek árkategóriájának kiíratása a szabad.txt fájlba");

                PrintWriter kiir = new PrintWriter(new FileWriter("szabad.txt"));

                for (i = 1; i <= sorDb; i++) {
                        for (j = 1; j <= helyDb; j++) {
                                if (foglaltsag[i - 1][j - 1].equals("x")) {
                                        kiir.print("x");
                                }
                                if (foglaltsag[i - 1][j - 1].equals("o")) {
                                        kiir.print(kategoria[i - 1][j - 1]);
                                }
                        }
                        kiir.println();
                }
                kiir.close();
                System.out.println("A szabad.txt fájl kiíratása megtörtént. \n");

        }

}
```

```
A 1. feladat megoldása
A foglaltsag.txt és a kategoria.txt fájlok beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
Egy bekért hely foglaltságának eldöntése
Kérem a sor számát (max 15): 12
Kérem a szék számát (max 20): 8
A kiválasztott hely még szabad.

A 3. feladat megoldása
Hány jegyet adtak el eddig?
Az előadásra eddig 187 darab jegyet adtak el, ez a nézőtér 62%-a.

A 4. feladat megoldása
Melyik árkategóriából adták el a legtöbb jegyet?
A legtöbb jegyet a(z) 2. árkategóriában értékesítették.

Az 5. feladat megoldása
Az aktuális bevétel meghatározása
A színház jelenlegi bevétele 593500 Ft lenne.

A 6. feladat megoldása
Egyedülálló üres helyek vizsgálata
A nézőtéren 35 db egyedülálló üres hely van.

A 7. feladat megoldása
A szabad helyek árkategóriájának kiíratása a szabad.txt fájlba
A szabad.txt fájl kiíratása megtörtént.


A szabad.txt szövegfájl

xx2x2x1x1x1x11xxxx2x
xxxxxxxxxxxxxxx222x
x2x2x1x1x1x1x1x2x2x2
22x2x1x1x1x1x1x22x2x
xxxxxxxxxxxxxxx2333
xxxxxxxxxxxxxxxxxxx
333332222112222233333
xxxxxxxxxx22xxxxxxxx
x44xxxxxxxxx3xxxxx44
xxxxxx3xxxxxxx3xx444
x54444433x3333444xxx
555555x4x4x4x4x5x5x5
5xxxxxxxxxxxxxxxxx55
xxxxxxxxxxxx4455555x
5xxxxxxxxxxxxx555555
```

2015 május: Expedíció

Valamikor a távközlés hőskorában egy ritka farkasfaj tudományos megfigyelésére expedíciót szerveztek a sarkkörön túlra. A magukkal vitt rádió csak napi egy adásra volt alkalmas, arra is csak 90 időegységig, időegységenként egy karaktert továbbítva. Az expedíció rádiósának üzeneteit több rádióamatőr is igyekezett lejegyezni. A feladatban a rádióamatőrök által lejegyzett üzeneteket kell feldolgoznia.

A *veetel.txt* fájl tartalmazza a rádióamatőrök által feljegyzett üzeneteket. Minden sorpár egy-egy feljegyzést tartalmaz.

- A sorpár első sorában két szám áll, az első a nap sorszáma, a második pedig – az előzőtől egy szóközzel elválasztva – a rádióamatőré.
- A sorpár második sorában a feljegyzéshez tartozó pontosan 90 karakter áll. A vett karakter az angol ábécé kisbetűje, számjegy, / jel vagy szóköz lehet. Ha az adott időegységben nem volt egyértelműen azonosítható a vett jel, akkor # karakter szerepel. Ha a tényleges üzenet befejeződött, az adó a fennmaradó időegységekben $ jelet küld.
- A napok sorszáma 1 és 11, a rádióamatőrök sorszáma 1 és 20 közötti egész szám lehet.
- Ha a megfigyelés során láttak farkasokat, akkor az üzenet két, / jellel elválasztott egész számmal, a látott kifejlett és kölyök egyedek számával kezdődik, amelyet szóköz követ. Más esetben nem szám az első karakter.

Például:

```
2 15
1/0 #gy#domb##l fig###tu# f#i#s ho#a##dalyoz$$...
```

A fenti sorpár első sora mutatja, hogy az üzenet a 2. napon érkezett és a 15-ös rádióamatőr rögzítette. 1 felnőtt és 0 kölyök farkast figyeltek meg. Mivel a második sorban a 45. karakter $ jel, és előtte nem # jel szerepel, ezért az üzenet biztosan 44 karakter hosszú.

Készítsen programot, amely a *veetel.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *radio* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja a *veetel.txt* fájl tartalmát!
2. Írja a képernyőre, hogy melyik rádióamatőr rögzítette az állományban szereplő első és melyik az utolsó üzenetet!
3. Adja meg az összes olyan feljegyzés napját és a rádióamatőr sorszámát, amelynek szövegében a „***farkas***” karaktersorozat szerepel!
4. Készítsen statisztikát, amely megadja, hogy melyik napon hány rádióamatőr készített feljegyzést. Azok a napok 0 értékkel szerepeljenek, amikor nem született feljegyzés! Az eredmény a képernyőn jelenjen meg a napok sorszáma szerint növekvően!
   A megjelenítést a feladat végén látható minta szerint alakítsa ki!

5. A rögzített üzenetek alapján kísérelje meg helyreállítani az expedíció által küldött üzenetet! Készítse el az *adaas.txt* fájlt, amely napok szerinti sorrendben tartalmazza a küldött üzeneteket! Ha egy időpontban senkinél nem volt vétel, akkor azon a ponton a # jel szerepeljen! (Feltételezheti, hogy az azonos üzenethez tartozó feljegyzések között nincs ellentmondás.)

   Az alábbi minta az első napról tartalmaz három üzenetet:

```
1 13
#abor# #e#tun###agy#szel#2# #o##h#d#g ##rkasn#o#oka# #a#tunk
e####a#akn##$#$#$##$$$$$$$$$
1 19
ta###t##ertunk ##gy #zel#####ok hide###f#r##sn#omo#at ##ttu##  e#y
patak#al$#$$$$$###$$$$$$$
1 9
ta#o#t#v##tu#k nag# #zel#20 fok#hi##g fa#k#snyo#okat la#tun#
#e#y#pat##na#$$###$$#$#$$$$$$$
```

   A helyreállított üzenet:

```
tabort vertunk nagy szel#20 fok hideg farkasnyomokat lattunk  e#y
pataknal$$$$$$$$$$$$$$$$$
```

6. Készítsen függvényt `szame` néven az alábbi algoritmus alapján! A függvény egy karaktersorozathoz hozzárendeli az igaz vagy a hamis értéket. A függvény elkészítésekor az algoritmusban megadott változóneveket használja! Az elkészített függvényt a következő feladat megoldásánál felhasználhatja.

```
Függvény szame(szo:karaktersorozat): logikai
  valasz:=igaz
  Ciklus i:=1-től hossz(szo)-ig
     ha szo[i]<'0' vagy szo[i]>'9' akkor valasz:=hamis
  Ciklus vége
  szame:=valasz
Függvény vége
```

7. Olvassa be egy nap és egy rádióamatőr sorszámát, majd írja a képernyőre a megfigyelt egyedek számát (a kifejlett és kölyök egyedek számának összegét)! Ha nem volt ilyen feljegyzés, a „Nincs ilyen feljegyzés” szöveget jelenítse meg! Ha nem volt megfigyelt egyed vagy számuk nem állapítható meg, a „Nincs információ” szöveget jelenítse meg! Amennyiben egy számot közvetlenül # jel követ, akkor a számot tekintse nem megállapíthatónak!

Minta a szöveges kimenetek kialakításához:

```
2. feladat:
Az első üzenet rögzítője: 13
Az utolsó üzenet rögzítője: 18

3. feladat:
10. nap 16. rádióamatőr
…

4. feladat:
1. nap: 13 rádióamatőr
2. nap: 14 rádióamatőr
…

7. feladat:
Adja meg a nap sorszámát! 2
Adja meg a rádióamatőr sorszámát! 15
A megfigyelt egyedek száma: 1
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Expedíció
 *
 * @author Klemand
 */

public class EmeltInfo2015maj {

	public static void main(String[] args) throws IOException {

		System.out.println("A 1. feladat megoldása");
		System.out.println("A veetel.txt fájl beolvasása ");

		BufferedReader beolvas = new BufferedReader(new FileReader("veetel.txt"));

		int[] nap = new int[220];
		int[] radios = new int[220];
		String[] uzenet = new String[220];
		int db = 0;

		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			daraboltSor = sor.split(" ");
			sor = beolvas.readLine();

			nap[db - 1] = Integer.parseInt(daraboltSor[0]);
			radios[db - 1] = Integer.parseInt(daraboltSor[1]);
			uzenet[db - 1] = sor;
		}

		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Az első üzenetet rögzítő rádiós: " + radios[0]);
		System.out.println("Az utolsó üzenetet rögzítő rádiós: " + radios[db - 1]);
```

```
System.out.println("\nA 3. feladat megoldása");
System.out.println("Mely napokon, mely rádiósok feljegyzésében szerepel a farkas szó?");

// Aadatokat napok, azon belül rádiósok szerint rendezzük.
// Először a rádiósok szerint rendezünk buborékos rendezéssel
int i, j;
int intAsztal;
String stringAsztal;

for (i = db - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (radios[j - 1] > radios[j]) {
                        intAsztal = nap[j - 1];
                        nap[j - 1] = nap[j];
                        nap[j] = intAsztal;

                        intAsztal = radios[j - 1];
                        radios[j - 1] = radios[j];
                        radios[j] = intAsztal;

                        stringAsztal = uzenet[j - 1];
                        uzenet[j - 1] = uzenet[j];
                        uzenet[j] = stringAsztal;
                }
        }
}

// Ezután a főszempont, a napok szerint rendezünk
// buborékos rendezéssel, ami a korábbi rendezettséget megőrzi.
for (i = db - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (nap[j - 1] > nap[j]) {
                        intAsztal = nap[j - 1];
                        nap[j - 1] = nap[j];
                        nap[j] = intAsztal;

                        intAsztal = radios[j - 1];
                        radios[j - 1] = radios[j];
                        radios[j] = intAsztal;

                        stringAsztal = uzenet[j - 1];
                        uzenet[j - 1] = uzenet[j];
                        uzenet[j] = stringAsztal;
                }
        }
}

for (i = 1; i <= db; i++) {
        if (uzenet[i - 1].contains("farkas")) {
                System.out.println(nap[i - 1] + ". nap " + radios[i - 1] + ". rádióamatőr");
        }
}
```

```
System.out.println("\nA 4. feladat megoldása");
System.out.println("Naponta hány amatőr készített feljegyzést?");

int napDb = 11;
int[] napokDb = new int[napDb];

for (i = 1; i <= napDb; i++) {
        napokDb[i - 1] = 0;
}

int aktNap;
for (i = 1; i <= db; i++) {
        aktNap = nap[i - 1];
        napokDb[aktNap - 1]++;
}

for (i = 1; i <= napDb; i++) {
        System.out.println(i + ". nap: " + napokDb[i - 1] + " rádióamatőr");
}

System.out.println("\nAz 5. feladat megoldása ");
System.out.println("A helyreállított üzenetek kiíratása az adaas.txt fájlba ");

PrintWriter kiir = new PrintWriter(new FileWriter("adaas.txt"));

String helyreallitott;
int n, k;
int karDb = 90;
int napKezd = 1; // egy adott nap első üzenetének sorszáma
int aktDb = 0;
;
for (n = 1; n <= napDb; n++) {
        helyreallitott = "";
        for (k = 1; k <= karDb; k++) {
                i = napKezd;
                aktDb = napokDb[n - 1];
                while (i <= (napKezd + aktDb - 1) && uzenet[i - 1].substring(k - 1, k).equals("#")) {
                        i++;
                }
                if (i <= napKezd + aktDb - 1) {
                        helyreallitott += uzenet[i - 1].substring(k - 1, k);
                } else {
                        helyreallitott += "#";
                }
        }
        kiir.println(helyreallitott);
        napKezd += aktDb;
}
kiir.close();
System.out.println("Az adaas.txt fájl kiíratása megtörtént. \n");

System.out.println("A 6. feladat megoldása ");
System.out.println("A szamE függvény elkészítése \n");
// A függvény a main metódus után található.
```

```
                System.out.println("A 7. feladat megoldása ");
                System.out.println("Egy bekért napon egy bekért rádiós által megfigyelt farkasok száma ");

                Scanner sc = new Scanner(System.in);
                System.out.print("Kérem a nap sorszámát (max. 11): ");
                aktNap = sc.nextInt();
                System.out.print("Kérem a rádiós sorszámát (max. 20): ");
                int aktRadios = sc.nextInt();

                String aktUzenet;
                int farkasDb = 0;
                int index;

                i = 1;
                while (i <= db && !(nap[i - 1] == aktNap && radios[i - 1] == aktRadios)) {
                        i++;
                }
                if (i > db) {
                        System.out.println("Nincs ilyen feljegyzés");
                } else {
                        aktUzenet = uzenet[i - 1];
                        index = aktUzenet.indexOf(" ");
                        aktUzenet = aktUzenet.substring(0, index);
                        if (aktUzenet.contains("/")) {
                                daraboltSor = aktUzenet.split("/");
                                if (szamE(daraboltSor[0]) && szamE(daraboltSor[1])) {
                                        farkasDb = Integer.parseInt(daraboltSor[0])
                                                        + Integer.parseInt(daraboltSor[1]);
                                        System.out.println("A megfigyelt egyedek száma: " + farkasDb);
                                } else {
                                        System.out.println("Nincs információ");
                                }
                        } else {
                                System.out.println("Nincs információ");
                        }
                }
                sc.close();
                System.out.println("");

        }

        public static boolean szamE(String szo) {
                boolean valasz = true;
                int i;
                for (i = 1; i <= szo.length(); i++) {
                        if (szo.charAt(i - 1) < '0' || szo.charAt(i - 1) > '9') {
                                valasz = false;
                        }
                }
                return valasz;
        }

}
```

```
A 1. feladat megoldása
A veetel.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
Az első üzenetet rögzítő rádiós: 13
Az utolsó üzenetet rögzítő rádiós: 18

A 3. feladat megoldása
Mely napokon, mely rádiósok feljegyzésében szerepel a farkas szó?
5. nap 15. rádióamatőr
10. nap 16. rádióamatőr

A 4. feladat megoldása
Naponta hány amatőr készített feljegyzést?
1. nap: 13 rádióamatőr
2. nap: 14 rádióamatőr
3. nap: 15 rádióamatőr
4. nap: 15 rádióamatőr
5. nap: 14 rádióamatőr
6. nap: 15 rádióamatőr
7. nap: 12 rádióamatőr
8. nap: 14 rádióamatőr
9. nap: 13 rádióamatőr
10. nap: 14 rádióamatőr
11. nap: 14 rádióamatőr

Az 5. feladat megoldása
A helyreállított üzenetek kiíratása az adaas.txt fájlba
Az adaas.txt fájl kiíratása megtörtént.

A 6. feladat megoldása
A szamE függvény elkészítése

A 7. feladat megoldása
Egy bekért napon egy bekért rádiós által megfigyelt farkasok száma
Kérem a nap sorszámát (max. 11): 2
Kérem a rádiós sorszámát (max. 20): 15
A megfigyelt egyedek száma: 1
```

Az adas.txt szövegfájl

```
tabort vertunk nagy szel 20 fok hideg farkasnyomokat lattunk egy pataknal$$$$$$$$$$$$$$$$
1/0 egy dombrol figyeltuk friss ho akadalyoz$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
2/3 rossz szelirany miatt csak rovid ideig$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
hovihar miatt nem volt megfigyeles rendberaktuk a felszerelest$$$$$$$$$$$$$$$$$$$$$$$$$$
a pataknal farkasok nem jelentkeztek nyomok voltak$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
0/3 a patakon tuli dombon kolykok jatszottak del korul$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
1/3 sotetedes elott tuntek fel az erdo szelen$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
verofenyes napfeny enyhulo ido$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
1/0 csak a nosteny jelent meg del korul$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
eszakrol gyakori farkasuvoltes hallatszik$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
13/0 sotetedes elott egy egesz falka haladt vegig a patak menten$$$$$$$$$$$$$$$$$$$$$$$$
```

2015 május idegennyelvű: Latin táncok

A Latin Tánciskola tanulói latin táncokat tanulnak, ezek a következők: cha-cha, salsa, rumba, samba, jive, tango, bachata.

A tanulók a tanév végén bemutatót tartottak. A bemutatón minden táncot csupán egyszer mutattak be, azonban az egyes táncok bemutatóján több pár is szerepelt. Az év végi bemutató táncrendjét a *tancrend.txt* fájl tartalmazza. A fájlban a táncok a bemutató tényleges sorrendjében szerepelnek. Tánconként minden párhoz három sor tartozik, ezek rendre a bemutatott táncot, majd a pár lány, végül a pár fiú tagjának utónevét tartalmazzák:

```
cha-cha
Katalin
Bertalan
cha-cha
Adrienn
Lajos
salsa
Katalin
Bertalan
```

A fenti példa szerint a cha-chát két pár, Katalin és Bertalan, valamint Adrienn és Lajos mutatták be, a cha-cha után pedig a salsa következett. Egy személy a különböző táncokat eltérő partnerekkel is bemutathatja, de feltételezheti, hogy a táncosok között nincs két azonos nevű.

A fájl legfeljebb 140 tánc és táncospár nevét tartalmazza, továbbá tudjuk, hogy legfeljebb 20 fiú, és legfeljebb 20 lány vett részt a bemutatón. Készítsen programot, amely a *tancrend.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *tanciskola* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be a *tancrend.txt* állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat!
2. Írassa ki a képernyőre, hogy melyik volt az elsőként és melyik az utolsóként bemutatott tánc neve!
3. Hány pár mutatta be a sambát? A választ jelenítse meg a képernyőn!
4. Írassa ki a képernyőre, hogy Vilma mely táncokban szerepelt!
5. Kérje be egy tánc nevét, majd írassa ki a képernyőre, hogy az adott táncot Vilma kivel mutatta be! Például ha a bekért tánc a samba, és Vilma párja Bertalan volt, akkor „`A samba bemutatóján Vilma párja Bertalan volt.`” szöveg jelenjen meg!
   Ha Vilma az adott tánc bemutatóján nem szerepelt, akkor azt írja ki a képernyőre, hogy „`Vilma nem táncolt samba-t.`”.

6. Készítsen listát a bemutatón részt vett fiúkról és lányokról! A listát a *szereplok.txt* nevű szöveges állományba mentse el a következő formátumban: a neveket vesszők válasszák el egymástól, de az utolsó név után már ne szerepeljen írásjel. Például:

```
Lányok: Lujza, Katalin, Andrea, Emma
Fiúk: Ferenc, Ambrus, Andor, Kelemen, Bertalan
```

7. Írja ki a képernyőre, hogy melyik fiú szerepelt a legtöbbször a fiúk közül, és melyik lány a lányok közül! Ha több fiú, vagy több lány is megfelel a feltételeknek, akkor valamennyi fiú, illetve valamennyi lány nevét írja ki!

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Latin táncok
 *
 * @author Klemand
 */

public class EmeltInfo2015mid {

	public static void main(String[] args) throws IOException {

		System.out.println("A 1. feladat megoldása");
		System.out.println("A tancrend.txt fájl beolvasása ");

		BufferedReader beolvas = new BufferedReader(new FileReader("tancrend.txt"));

		String[] tanc = new String[140];
		String[] lany = new String[140];
		String[] fiu = new String[140];
		int db = 0;
		String sor;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			tanc[db - 1] = sor;
			lany[db - 1] = beolvas.readLine();
			fiu[db - 1] = beolvas.readLine();
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása ");
		System.out.println("Az elsőként bemutatott tánc neve: " + tanc[0]);
		System.out.println("Az utolsóként bemutatott tánc neve: " + tanc[db - 1]);

		System.out.println("\nA 3. feladat megoldása ");
		System.out.println("Hány pár mutatta be a sambát?");

		int sambaDb = 0;
		int i;
		for (i = 1; i <= db; i++) {
			if (tanc[i - 1].equals("samba")) {
				sambaDb++;
			}
		}
		System.out.println("A sambát " + sambaDb + " pár mutatta be. \n");

		System.out.println("A 4. feladat megoldása ");
		System.out.print("Vilma a következő táncokban szerepelt: ");

		for (i = 1; i <= db; i++) {
			if (lany[i - 1].equals("Vilma")) {
				System.out.print(tanc[i - 1] + " ");
			}
		}
```

```
System.out.println("\n\nAz 5. feladat megoldása ");
System.out.println("Egy bekért táncot kivel táncolt Vilma? ");

Scanner sc = new Scanner(System.in);
System.out.print("Kérem a tánc nevét: ");
String aktTanc = sc.nextLine();

i = 1;
while (i <= db && !(tanc[i - 1].equals(aktTanc) && lany[i - 1].equals("Vilma"))) {
        i++;
}
if (i <= db) {
        System.out.println("A " + aktTanc + " bemutatóján Vilma párja " + fiu[i - 1] + " volt.");
} else {
        System.out.println("Vilma nem táncolt " + aktTanc + "-t.");
}
sc.close();

System.out.println("\nA 6. feladat megoldása ");
System.out.println("A bemutatón részt vevők listájának kiíratása a szereplok.txt fájlba");

// A lányok és a fiúk listála és táncaik száma
String[] lanyLista = new String[20];
int lanyListaDb = 0;
int[] lanyTancDb = new int[20];
String[] fiuLista = new String[20];
int fiuListaDb = 0;
int[] fiuTancDb = new int[20];

int j;
String aktLany, aktFiu;
for (i = 1; i <= db; i++) {
        aktLany = lany[i - 1];
        j = 1;
        while (j <= lanyListaDb && !(lanyLista[j - 1].equals(aktLany))) {
                j++;
        }
        if (j <= lanyListaDb) {
                lanyTancDb[j - 1]++;
        } else {
                lanyListaDb++;
                lanyLista[lanyListaDb - 1] = aktLany;
                lanyTancDb[lanyListaDb - 1] = 1;
        }

        aktFiu = fiu[i - 1];
        j = 1;
        while (j <= fiuListaDb && !(fiuLista[j - 1].equals(aktFiu))) {
                j++;
        }
        if (j <= fiuListaDb) {
                fiuTancDb[j - 1]++;
        } else {
                fiuListaDb++;
                fiuLista[fiuListaDb - 1] = aktFiu;
                fiuTancDb[fiuListaDb - 1] = 1;
        }
}
```

```
                PrintWriter kiir = new PrintWriter(new FileWriter("szereplok.txt"));

                kiir.print("Lányok: ");
                for (i = 1; i < lanyListaDb; i++) {
                        kiir.print(lanyLista[i - 1] + ", ");
                }
                kiir.println(lanyLista[lanyListaDb - 1]);

                kiir.print("Fiúk: ");

                for (i = 1; i < fiuListaDb; i++) {
                        kiir.print(fiuLista[i - 1] + ", ");
                }
                kiir.println(fiuLista[fiuListaDb - 1]);

                kiir.close();
                System.out.println("A szereplok.txt fájl kiíratása megtörtént. \n");

                System.out.println("A 7. feladat megoldása ");
                System.out.println("A legtöbbször szereplők nemenként:");

                int lanyMaxDb = 0;
                for (i = 1; i < lanyListaDb; i++) {
                        if (lanyTancDb[i - 1] > lanyMaxDb) {
                                lanyMaxDb = lanyTancDb[i - 1];
                        }
                }

                System.out.print("Lányok:");

                for (i = 1; i < lanyListaDb; i++) {
                        if (lanyTancDb[i - 1] == lanyMaxDb) {
                                System.out.print(" " + lanyLista[i - 1]);
                        }
                }

                int fiuMaxDb = 0;
                for (i = 1; i < fiuListaDb; i++) {
                        if (fiuTancDb[i - 1] > fiuMaxDb) {
                                fiuMaxDb = fiuTancDb[i - 1];
                        }
                }

                System.out.println("");
                System.out.print("Fiúk:");

                for (i = 1; i < fiuListaDb; i++) {
                        if (fiuTancDb[i - 1] == fiuMaxDb) {
                                System.out.print(" " + fiuLista[i - 1]);
                        }
                }
                System.out.println("\n");

        }

}
```

```
A 1. feladat megoldása
A tancrend.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
Az elsőként bemutatott tánc neve: cha-cha
Az utolsóként bemutatott tánc neve: bachata

A 3. feladat megoldása
Hány pár mutatta be a sambát?
A sambát 10 pár mutatta be.

A 4. feladat megoldása
Vilma a következő táncokban szerepelt: cha-cha rumba tango

Az 5. feladat megoldása
Egy bekért táncot kivel táncolt Vilma?
Kérem a tánc nevét: tango
A tango bemutatóján Vilma párja Tivadar volt.

A 6. feladat megoldása
A bemutatón részt vevők listájának kiíratása a szereplok.txt fájlba
A szereplok.txt fájl kiíratása megtörtént.

A 7. feladat megoldása
A legtöbbször szereplők nemenként:
Lányok: Katalin Szilvia
Fiúk: Kelemen Bertalan
```

A szereplok.txt szövegfájl

```
Lányok: Luca, Katalin, Adrienn, Emma, Vilma, Szilvia, Elza, Fruzsina, Judit, Gabriella, Irma, Szabina
Fiúk: Kelemen, Bertalan, Lajos, Igor, Kolos, Alfonz, Tivadar, Ede, Imre, Attila, Salamon, Marcell
```

2015 október: Fej vagy írás

Ha egy szabályos pénzérmét feldobunk, ugyanannyi a valószínűsége annak, hogy leesés után az érme értéke lesz felül (írás, I), mint annak, hogy a címert tartalmazó másik oldala (fej, F). Ezért gyakran „pénzfeldobással” sorsolnak, például így döntik el, hogy melyik csapat kezdhet el egy futballmeccset.

Feladata a pénzfeldobás szimulálása, illetve pénzfeldobással kapott sorozatok elemzése lesz. A feladatok során az írást az I, a fejet az F nagybetű jelzi. Például egy 5 feldobásból álló sorozat esetén:

```
I
I
F
I
F
```

Készítsen programot *fejvagyiras* néven a következő feladatok megoldására! A program futása során a képernyőre való kiíráskor, illetve az adatok billentyűzetről való beolvasásakor utaljon a feladat sorszámára és a kiírandó, illetve bekérendő adatra! Az ékezetmentes kiírás is elfogadott.

1. Szimuláljon egy pénzfeldobást, ahol azonos esélye van a fejnek és az írásnak is! Az eredményt írassa ki a képernyőre a mintának megfelelően!
2. Kérjen be a felhasználótól egy tippet, majd szimuláljon egy pénzfeldobást! Írassa ki a képernyőre a felhasználó tippjét és a dobás eredményét is, majd tájékoztassa a felhasználót az eredményről következő formában: „`Ön eltalálta.`” vagy „`Ön nem találta el.`”!

A *kiserlet.txt* állományban egy pénzfeldobás-sorozat eredményét találja. Mivel a sorozat hossza tetszőleges lehet, ezért az **összes adat memóriában történő egyidejű eltárolása nélkül** oldja meg a következő feladatokat! Feltételezheti, hogy egymilliónál több adata nem lesz.

3. Állapítsa meg, hány dobásból állt a kísérlet, és a választ a mintának megfelelően írassa ki a képernyőre!
4. Milyen relatív gyakorisággal dobtunk a kísérlet során fejet? (A fej relatív gyakorisága a fejet eredményező dobások és az összes dobás hányadosa.) A relatív gyakoriságot a mintának megfelelően két tizedesjegy pontossággal, százalék formátumban írassa ki a képernyőre!
5. Hányszor fordult elő ebben a kísérletben, hogy egymás után pontosan két fejet dobtunk?
   A választ a mintának megfelelően írassa ki a képernyőre!
   (Feltételezheti, hogy a kísérlet legalább 3 dobásból állt.)

   Például az `IFFFFIIFFIFFFIFF` sorozatban kétszer fordult elő, hogy egymás után pontosan két fejet dobtunk.
6. Milyen hosszú volt a leghosszabb, csak fejekből álló részsorozat? Írassa ki a választ a képernyőre a mintának megfelelően, és adja meg egy ilyen részsorozat első tagjának helyét is!
   (A minta tagjainak számozását eggyel kezdjük.)

Sokan azt hiszik, hogy ha már elég sok fejet dobtunk, akkor a következő dobás nagyobb valószínűséggel lesz írás, mint fej. Ennek ellenőrzésére vonatkozik a következő feladat.

7. Állítson elő és tároljon a memóriában 1000 db négy dobásból álló sorozatot! Számolja meg, hogy hány esetben követett egy háromtagú „tisztafej” sorozatot fej, illetve hány esetben írás! Az eredményt írassa ki a *dobasok.txt* állományba úgy, hogy az első sorba kerüljön az eredmény, a második sorban pedig egy-egy szóközzel elválasztva, egyetlen sorban szerepeljenek a dobássorozatok!

   Például:

```
FFFF: 12, FFFI: 14
FIFI IIIF IFIF IIII FFII FFFF IIFI FFII FFFI …
```

Minta (a forrásállomány alapján készült, valós adatokat tartalmaz):

```
1. feladat
A pénzfeldobás eredménye: I
2. feladat
Tippeljen! (F/I)= I
A tipp I, a dobás eredménye I volt.
Ön eltalálta!
3. feladat
A kísérlet 4321 dobásból állt.
4. feladat
A kísérlet során a fej relatív gyakorisága 51,03% volt.
5. feladat
A kísérlet során 259 alkalommal dobtak pontosan két fejet egymás után.
6. feladat
A leghosszabb tisztafej sorozat 11 tagból áll, kezdete a(z) 947. dobás.
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Fej vagy írás
 *
 * @author Klemand
 */

public class EmeltInfo2015okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása ");
		System.out.println("Pénzfeldobás szimulálása");
		String dobas = dobasSzim();
		System.out.println("A pénzfeldobás eredménye: " + dobas);

		System.out.println("\nA 2. feladat megoldása ");
		System.out.println("Egy felhasználói tippelés eredménye");

		Scanner sc = new Scanner(System.in);
		System.out.print("Tippeljen! (F/I) = ");
		String tipp = sc.nextLine().toUpperCase();
		// Kisbetűs beírás esetén is működjön!

		dobas = dobasSzim();
		System.out.println("A tipp " + tipp + ", a dobás eredménye " + dobas + " volt.");

		if (dobas.equals(tipp)) {
			System.out.println("Ön eltalálta.");
		} else {
			System.out.println("Ön nem találta el.");
		}
		sc.close();

		/*
		 * A következő feladatokat a kiserlet.txt fájl alapján,
		 * de annak memóriában történő teljes tárolása nélkül kell megoldani,
		 * azaz egyszerre csak néhány sor lehet a memóriában!
		 */

		// A 3. és a 4. feladatot célszerű egy beolvasással megoldani.
		BufferedReader beolvas = new BufferedReader(new FileReader("kiserlet.txt"));

		int db = 0;
		int fejDb = 0;
		String sor;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			if (sor.equals("F")) {
				fejDb++;
			}
		}
```

```
System.out.println("\nA 3. feladat megoldása");
System.out.println("A kísérlet " + db + " dobásból állt.\n");

System.out.println("A 4. feladat megoldása");
double szazalek = ((double) fejDb / db) * 100;
System.out.print("A kísérlet során a fej relatív gyakorisága ");
System.out.printf("%.2f", szazalek);
System.out.println("% volt.\n");
beolvas.close();

System.out.println("Az 5. feladat megoldása ");

beolvas = new BufferedReader(new FileReader("kiserlet.txt"));

int ffDb = 0;
String aktSor;
aktSor = "I";
aktSor += beolvas.readLine();
aktSor += beolvas.readLine();

int i;
for (i = 3; i <= db + 1; i++) {
        aktSor += beolvas.readLine();
        if ((aktSor.equals("IFFI"))) {
                ffDb++;
        }
        aktSor = aktSor.substring(1);
}
if ((aktSor.equals("IFF"))) {
        ffDb++;
}
beolvas.close();

System.out.print("A kísérlet során " + ffDb);
System.out.println(" alkalommal dobtak pontosan két fejet egymás után. \n");

System.out.println("A 6. feladat megoldása ");
System.out.println("A leghosszabb tisztafej sorozat");

beolvas = new BufferedReader(new FileReader("kiserlet.txt"));

int aktTkezdet = -1; // irreális kezdőérték
int aktThossz = 0;
int maxTkezdet = -1; // irreális kezdőérték
int maxThossz = 0;

for (i = 1; i <= db; i++) {
        sor = beolvas.readLine();

        if (sor.equals("F")) {
                // Ez a T tulajdonság
                if (aktTkezdet == -1) {
                        aktTkezdet = i;
                }
                aktThossz++;
                if (aktThossz > maxThossz) {
                        maxTkezdet = aktTkezdet;
                        maxThossz = aktThossz;
                }
        } else {
                aktTkezdet = -1;
                aktThossz = 0;
        }
}
```

```
                if (maxTkezdet > -1) {
                        System.out.print("A leghosszabb tisztafej sorozat " + maxThossz + " tagból áll, ");
                        System.out.println("kezdete: " + maxTkezdet + ". dobás.");
                } else {
                        System.out.println("A sorozatban nincs fej.");
                }
                beolvas.close();

                System.out.println("\nA 7. feladat megoldása ");
                System.out.println("Egy szimulált kísérletsorozat eredményének kiíratása a dobasok.txt fájlba");

                String[] sorozat = new String[1000];
                int ffffDb = 0;
                int fffiDb = 0;

                for (i = 1; i <= 1000; i++) {
                        sorozat[i - 1] = sorozatSzim();

                        if (sorozat[i - 1].equals("FFFF")) {
                                ffffDb++;
                        }
                        if (sorozat[i - 1].equals("FFFI")) {
                                fffiDb++;
                        }
                }

                PrintWriter kiir = new PrintWriter(new FileWriter("dobasok.txt"));

                kiir.println("FFFF: " + ffffDb + ", FFFI: " + fffiDb);
                for (i = 1; i <= 1000; i++) {
                        kiir.print(sorozat[i - 1] + " ");
                }
                kiir.close();
                System.out.println("A dobasok.txt fájl kiíratása megtörtént. \n");

        }

        public static String dobasSzim() {
                double veletlenSzam = Math.random();
                if (veletlenSzam < 0.5) {
                        // Az eredmény a [0;1[ intervallumba eshet
                        return "F";
                } else {
                        return "I";
                }
        }

        public static String sorozatSzim() {
                int i;
                String sor = "";
                for (i = 1; i <= 4; i++) {
                        sor += dobasSzim();
                }
                return sor;
        }

}
```

```
Az 1. feladat megoldása
Pénzfeldobás szimulálása
A pénzfeldobás eredménye: I

A 2. feladat megoldása
Egy felhasználói tippelés eredménye
Tippeljen! (F/I) = f
A tipp F, a dobás eredménye I volt.
Ön nem találta el.

A 3. feladat megoldása
A kísérlet 4321 dobásból állt.

A 4. feladat megoldása
A kísérlet során a fej relatív gyakorisága 51,03% volt.

Az 5. feladat megoldása
A kísérlet során 259 alkalommal dobtak pontosan két fejet egymás után.

A 6. feladat megoldása
A leghosszabb tisztafej sorozat
A leghosszabb tisztafej sorozat 11 tagból áll, kezdete: 947. dobás.

A 7. feladat megoldása
Egy szimulált kísérletsorozat eredményének kiíratása a dobasok.txt fájlba
A dobasok.txt fájl kiíratása megtörtént.


A dobasok.txt szövegfájl

FFFF: 81, FFFI: 58
FFFI FIFI FFII FIIF IIII IFII FFIF FFFI IIFF IFIF IIII FFFF IIII IIIF IIFF IFFI FFIF IIFF FIII IIII IFFF FIII FFIF
FIIF FFIF FIIF FIFI FIIF IIFI IFII IIIF FFFF IFII FIFF IFFF IFIF IIFF FFFF FIIF IFFF IIIF IIFF FIII FFFF FFFF FFIF
IIFF FIFF IIIF IIIF FFFF FIFI FIII FFII FFFF FIFI FFFI FFIF FIIF IFII FFIF IFFF IFIF FIIF IFFI FIFF IIIF IFFI FFFF
IIFI FFFF FIIF IFIF IFFI IIIF FIFI IFIF IIFF FIII IFFF FFIF FFII IIII IFII FFFF IFFI FIIF FFFF FFII IFFI IFIF FFII
FIFF IFIF IFIF IIII FFFI IFII FIII IIIF FFFF IFII FIIF IFII IFIF IFFI IFFI IIIF IFII IFIF IFII IFFF FFFF FFIF FIII
FIIF IIII FIIF FIII IFFF IFII IIFF IIIF FIFF FIFF FIII IFFF IIFI IIIF FFFI IFIF FFIF FFFI IFFI FFIF IIII IFIF IIFI
FFFF FFII IIFF IIFF IIFI IIII FFIF IFFF FFFF IIFF FFFF FIII FIFF FFFI IIII FIIF IIIF FIFF IFII IIFF FFFF FFII FIFI
IFFI IFIF IIFI IIFI FFFF IFFI IIII IIFI FFFI IFFF FFFI FIFF IFII IIFI FFFF FIII FFFF IIIF FIIF FFIF IFFF IIIF IFII
FFFF IIIF IIFF IFIF IFIF FIFI FFIF FFFI IFFI FIFF FFFI IIFI IIII IIIF IFIF FIFI IFFF FFIF IFIF IFFF FFII FIFF FFFF
FFFF IFIF FIFF FIIF IFII IFIF FFIF FIIF IIIF IFFF IFFI IFIF IFIF IIFF IFFI IIIF IFFF FIII FFIF IFFF FFFF IFIF IFFI
FFFF IFII FFFI IIIF IFFF IIFF IIFF FFFI FIII FFII IIFI FFII IFFF FFII FIFI FIFI FFII FIFI FIII FFFF IIII IFFI IFII
IIFF IIIF IIII FIII IFII IIIF FIFI IFFI FIFI FFFI FFFI FIFF FIFI FFFF IIFF FIFI IFFI IIII FFIF IFIF IFFF FIFI FIFI
FIFF FFIF IIFI FFFF IFFI IFFI FIFI IIFI IIII IFIF FIIF FFIF IIFF FFII FFFI IIIF FFII FIII FIFI FFII FIIF FIFI IIFI
FIII IIIF FIII IFFF FFII FFFF IFII FIFI IFFI IFIF IIIF IIFI FFII IIIF IIFI FIFF FFII IFIF FFFF IIIF FIIF IFIF FFIF
IIFF IIFF FFFF FFFI FFFF IFFF IFFI IFFI IIFF IFII IIFF FFII IIIF FFII FFFI FFFI FFFF FIFF IIFF FIFI IIIF IFII FIFF
FIFF IIII FIFF FFIF IIFI FIFI IFIF IIII IFFF IIFI FFFF FFFI IFIF FIIF IFII FIIF IIFI FFFI IIFI IIII FIFI IFFF FIIF
IFFF IFIF FIFI IFIF FIII IIFI FFFF FFIF IIII IIIF IFII IFFI IFFF IFFF IIFF IIII FFIF FIFF IFFI FIFI FIFI IFII FIFI
FIII FIII FIII IFFF FFFF IIIF FFIF FIFF IIFI FIFI FIFI FIFF FFFI FFII IFFF IFII FFFI FFFF IFIF FFFI IIFF IIII IIFI
FIIF IIFI IFFF FFFI IFII FIIF IIII IIIF FFFI IIII FFIF FFII FIII FIFI IFFF FIFI IIIF IFFI IFFF IFFF IIFF IFIF IFFI
FFII IIFF FFIF IIIF IIIF IFII IIII IFIF FFFI IFII FIIF FFII IIFI IIIF FFFF FFFI IIFF FIFF IIFI IIFI FIFF IFIF IFFI
IFFI FFII IFFI FIIF FIII IIIF FFII FIFI IFFF IIFI FIII IIII FIII FIIF FIII FFFI IFFF FIFI IIIF FFFI IIIF FFII FIII
IFII FIFF FFII IFFI IIFI IFII FIFF FFFF FFFF FFII FFFI FIFF IIIF IIII IIII FFII FIII FIII IIFF FFFF IIFI FIFI IIIF
IIIF FIII FIFI IIFI FIII FFFF IFIF FIIF IIFF FIII IFFF IFII FFII FIFI FIFI FIFF IFFF IFFI FFFF FIIF IIFF IIII IFFI
IIFI FIFF FIII FIII FFII IFIF IIIF IFFF IIII IIIF FFII FFIF FIFF FIFF FIIF IFIF IIIF IFFI IFFI FIFF IFII FIIF IFFF
FIII FIFI IFII IIFI FFFF IFFI FIFI FIIF FFII IFFF IIIF FFFI FFIF IFII IFII IFII FIFF FIFI FFFF IIII FIII FFFF FFFF
FFII FIFF FIFI IFII FFFI FIIF FIFF IIFF FFII FFII IIII IIII IFFI IIIF IIIF IFIF IIFI FFFF FFIF IFIF FFIF FIII IFII
IFFI IIIF FIFI IIIF IFII IIIF IFFI FIII IIFI IIII FFII IFFF IIII FFIF FIII FFFI FIIF IFFF FFFI IIFF IFIF IFII FFFF
IIII FFIF IFFI IIFF IFFF FIII IIFI FFFF IFFI IFFI IFFF IFII IIII IFFF IIIF FIFF FFII FIIF IIII IFFF IFFF IFII IFFI
IFIF IFFF FIII IFIF IIFF FIFF IFFI IFIF FIIF IFII IFII FIIF FIIF IFFI IIFF IIFI IFFF FIII FFFF FIII FFIF IIFI IFIF
IFFF IIII IIFI IFIF FIII FIIF FFIF FFII IIIF FIII FFII IIFI IIIF FFFI FIIF FFII FFFI FIIF FFFF FFFF IFFI FFFF FFII
IFII FFII IIFF IIII FFFI IIFF FFII IFII FIIF IIII IFFI IFFI IFFI FIII FIFI IFIF IIII FIIF FFFF FIFF FFIF FIII IFFF
FFFF IFII IIIF IIII IFFF IFFI IFFI FFFF FIII FFFF IIFF FIII IIII FIII IFII FFFF IIII IIFI FIII IIFI FIFF IIIF FIIF
IIII IFFF IIIF IFFI FIIF FIFI IFIF FFII IFII IIIF FFIF FIFI IIIF FIII FFIF IFFF FFIF IFIF FIFI FFII IIII FIFI FIFI
...
```

2016 május: Ötszáz

Egy apróságokat árusító boltban minden árucikk darabja 500 Ft. Ha egy vásárlás során valaki egy adott árucikkből több darabot is vesz, a második ára már csak 450 Ft, a harmadik pedig 400 Ft, de a negyedik és további darabok is ennyibe kerülnek, tehát az ár a harmadik ugyanazon cikk vásárlása után már nem csökken tovább.

A pénztárhoz menők kosarában legalább 1 és legfeljebb 20 darab árucikk lehet. A kosarak tartalmát a *penztar.txt* fájl írja le, amelyben soronként egy-egy árucikk neve vagy az F karakter szerepel. A fájlban legfeljebb 1000 sor lehet. Az F karakter azt jelzi, hogy az adott vásárlónak nincs már újabb árucikk a kosarában, fizetés következik. Az árucikkek neve ékezet nélküli, több szóból is állhat, hossza legfeljebb 30 karakter.

Példa a *penztar.txt* fájl első néhány sorára:

```
toll
F
colostok
HB                          ceruza
HB                          ceruza
colostok
toll
szatyor
csavarkulcs
doboz
F
```

A példa alapján az első vásárló összesen 1 tollat vásárolt, ezért összesen 500 Ft-ot kell fizetnie. A második vásárlás során hatféle árucikket vásároltak – a HB ceruzából és a colostokból többet is –, összesen 3900 Ft értékben.

Készítsen programot, amely a *penztar.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *otszaz* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, és feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el a *penztar.txt* fájl tartalmát!
2. Határozza meg, hogy hányszor fizettek a pénztárnál!
3. Írja a képernyőre, hogy az első vásárlónak hány darab árucikk volt a kosarában!
4. Kérje be a felhasználótól egy vásárlás sorszámát, egy árucikk nevét és egy darabszámot! A következő három feladat megoldásánál ezeket használja fel!

   Feltételezheti, hogy a program futtasásakor csak a bemeneti állományban rögzített adatoknak megfelelő vásárlási sorszámot és árucikknevet ad meg a felhasználó.

5. Határozza meg, hogy a bekért árucikkből
    a. melyik vásárláskor vettek először, és melyiknél utoljára!
    b. összesen hány alkalommal vásároltak!
6. Határozza meg, hogy a bekért darabszámot vásárolva egy termékből mennyi a fizetendő összeg! A feladat megoldásához készítsen függvényt *ertek* néven, amely a darabszámhoz a fizetendő összeget rendeli!
7. Határozza meg, hogy a bekért sorszámú vásárláskor mely árucikkekből és milyen mennyiségben vásároltak! Az árucikkek nevét tetszőleges sorrendben megjelenítheti.
8. Készítse el az *osszeg.txt* fájlt, amelybe soronként az egy-egy vásárlás alkalmával fizetendő összeg kerüljön a kimeneti mintának megfelelően!

Minta a szöveges kimenetek kialakításához:

```
2. feladat
A fizetések száma: 141

3. feladat
Az első vásárló 1 darab árucikket vásárolt.

4. feladat
Adja meg egy vásárlás sorszámát! 2
Adja meg egy árucikk nevét! kefe
Adja meg a vásárolt darabszámot! 2

5. feladat
Az első vásárlás sorszáma: 5
Az utolsó vásárlás sorszáma: 139
32 vásárlás során vettek belőle.

6. feladat
2 darab vételekor fizetendő: 950

7. feladat
1 toll
1 szatyor
1 doboz
1 csavarkulcs
2 colostok
2 HB ceruza
```

Részlet az *osszeg.txt* fájlból:

```
1: 500
2: 3900
3: 2300
4: 1000
5: 2500
6: 2900
7: 950
…
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Ötszáz
 *
 * @author Klemand
 */

public class EmeltInfo2016maj {

        public static void main(String[] args) throws IOException {

                System.out.println("Az 1. feladat megoldása");
                System.out.println("A penztar.txt fájl beolvasása");

                BufferedReader beolvas = new BufferedReader(new FileReader("penztar.txt"));

                int[] vasarlo = new int[1000];
                int vasarloDb = 1;
                String[] arucikk = new String[1000];
                int db = 0;
                String sor;

                while ((sor = beolvas.readLine()) != null) {
                        if (sor.equals("F")) {
                                vasarloDb++;
                        } else {
                                db++;
                                arucikk[db - 1] = sor;
                                vasarlo[db - 1] = vasarloDb;
                        }
                }
                beolvas.close();
                vasarloDb--;// Az utolsó fizetésnél is növeltük!

                System.out.println("A beolvasás megtörtént.\n");

                System.out.println("A 2. feladat megoldása");
                System.out.println("A fizetések száma: " + vasarloDb);

                System.out.println("\nA 3. feladat megoldása");
                System.out.print("Az első vásárló ");

                int i = 0; //A darabszám miatt 0 a kezdőértéke!
                while (i <= db && vasarlo[i] == 1) {
                        i++;
                }
                System.out.println(i + " darab árucikket vásárolt. \n");

                System.out.println("A 4. feladat megoldása");
                Scanner sc1 = new Scanner(System.in);
                Scanner sc2 = new Scanner(System.in);

                int aktVasarlo;
                String aktArucikk;
                int aktDb;
```

```
System.out.print("Adja meg a vásárlás sorszámát (max. " + vasarloDb + "): ");
aktVasarlo = sc1.nextInt();
System.out.print("Adja meg az árucikk nevét: ");
aktArucikk = sc2.nextLine();
System.out.print("Adja meg a vásárolt darabszámot (max. 20): ");
aktDb = sc1.nextInt();
sc1.close();
sc2.close();

System.out.println("\nAz 5. feladat megoldása");
System.out.print("A bekért árucikkből az első vásárlás sorszáma: ");

i = 1;
// Kiválasztás, tudjuk, hogy létezik ilyen árucikk
while (!(arucikk[i - 1].equals(aktArucikk))) {
        i++;
}
System.out.println(vasarlo[i - 1]);

System.out.print("Az utolsó vásárlás sorszáma: ");

i = db;
// Kiválasztás, tudjuk, hogy létezik ilyen árucikk
while (!(arucikk[i - 1].equals(aktArucikk))) {
        i--;
}
System.out.println(vasarlo[i - 1]);

// A vásárlások listája vásárlónként és termékenként
// Ki, mit és abból mennyit vásárolt
int[] vasarloLista = new int[1000];
String[] arucikkLista = new String[1000];
int[] mennyiseg = new int[1000];
int vasarloListaDb = 0;
int v; // vásárló
String a; // árucikk
int j;
for (i = 1; i <= db; i++) {
        v = vasarlo[i - 1];
        a = arucikk[i - 1];
        j = 1;
        while (j <= vasarloListaDb && !(vasarloLista[j - 1] == v && arucikkLista[j - 1].equals(a))) {
                j++;
        }
        if (j <= vasarloListaDb) {
                mennyiseg[j - 1]++;
        } else {
                vasarloListaDb++;
                vasarloLista[vasarloListaDb - 1] = v;
                arucikkLista[vasarloListaDb - 1] = a;
                mennyiseg[vasarloListaDb - 1] = 1;
        }
}

int aktArucikkVasarlas = 0;
// Ennyi vásárlás során vettek egy vagy több darabot a bekért árucikkből!
for (i = 1; i <= vasarloListaDb; i++) {
        if (arucikkLista[i - 1].equals(aktArucikk)) {
                aktArucikkVasarlas++;
        }
}
System.out.println("Összesen " + aktArucikkVasarlas + " vásárlás során vettek belőle.\n");
```

```
                System.out.println("A 6. feladat megoldása");
                System.out.println(aktDb + " darab vételekor fizetendő: " + ertek(aktDb) + " Ft.\n");

                System.out.println("A 7. feladat megoldása");
                System.out.println("A bekért sorszámú vásárlás adatai: ");

                for (i = 1; i <= vasarloListaDb; i++) {
                        if (vasarloLista[i - 1] == aktVasarlo) {
                                System.out.println(mennyiseg[i - 1] + " " + arucikkLista[i - 1]);
                        }
                }

                System.out.println("");

                System.out.println("A 8. feladat megoldása");
                System.out.println("A vásárlások alkalmával fizetendő összegek kiírása az osszeg.txt fájlba.");

                PrintWriter kiir = new PrintWriter(new FileWriter("osszeg.txt"));

                int osszeg;
                for (i = 1; i <= vasarloDb; i++) {
                        osszeg = 0;
                        for (j = 1; j <= vasarloListaDb; j++) {
                                if (vasarloLista[j - 1] == i) {
                                        osszeg += ertek(mennyiseg[j - 1]);
                                }
                        }
                        kiir.println(i + ": " + osszeg);
                }
                kiir.close();
                System.out.println("A fájlkiírás megtörtént. \n");

        }

        public static int ertek(int db) {
                switch (db) {
                case 1:
                        return 500;
                case 2:
                        return 950;
                default:
                        return 950 + (db - 2) * 400;
                }
        }

}
```

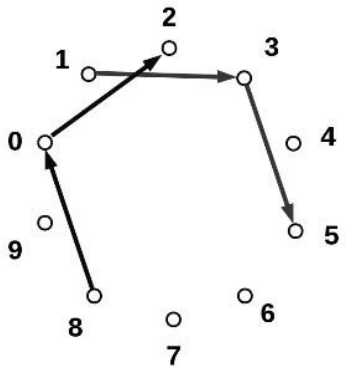
```
Az 1. feladat megoldása
A penztar.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
A fizetések száma: 141

A 3. feladat megoldása
Az első vásárló 1 darab árucikket vásárolt.

A 4. feladat megoldása
Adja meg a vásárlás sorszámát (max. 141): 2
Adja meg az árucikk nevét: kefe
Adja meg a vásárolt darabszámot (max. 20): 2

Az 5. feladat megoldása
A bekért árucikkből az első vásárlás sorszáma: 5
Az utolsó vásárlás sorszáma: 139
Összesen 32 vásárlás során vettek belőle.

A 6. feladat megoldása
2 darab vételekor fizetendő: 950 Ft.

A 7. feladat megoldása
A bekért sorszámú vásárlás adatai:
2 colostok
2 HB ceruza
1 toll
1 szatyor
1 csavarkulcs
1 doboz

A 8. feladat megoldása
A vásárlások alkalmával fizetendő összegek kiírása az osszeg.txt fájlba.
A fájlkiírás megtörtént.
```

Az osszeg.txt szövegfájl

```
1: 500
2: 3900
3: 2300
4: 1000
5: 2500
6: 2900
7: 950
8: 1950
9: 3750
10: 1500
11: 500
12: 1950
13: 500
14: 500
15: 1000
16: 500
17: 500
18: 1000
19: 1000
20: 1950
21: 500
22: 500
23: 1000
24: 1450
25: 500
26: 4750
27: 500
28: 1500
29: 1450
30: 1000
31: 500
32: 1000
33: 1500
34: 1950
35: 1000
36: 500
37: 1000
38: 1000
39: 1500
40: 500
41: 500
42: 2500
43: 1500
44: 500
45: 1500
46: 2450
47: 1500
48: 500
49: 1000
50: 1000
51: 500
52: 1500
53: 1000
54: 500
55: 500
56: 500
57: 500
58: 1000
59: 1500
60: 500
61: 2500
62: 500
63: 500
64: 1000
65: 1000
66: 1000
67: 1000
68: 500
69: 950
70: 2900
71: 1000
72: 500
73: 2450
74: 2450
75: 2000
76: 1450
77: 2400
78: 950
79: 3450
80: 500
81: 1950
82: 500
83: 1000
84: 1000
85: 500
86: 500
87: 1000
88: 500
89: 1500
90: 500
91: 500
92: 1000
93: 1500
94: 3450
95: 1500
96: 3800
97: 1000
98: 2000
99: 1000
100: 500
...
```

2016 május idegennyelvű: Zár

Egy ajtót elektronikus zárral láttak el. A zárat egy ismétlődő pontokat nem tartalmazó, megfelelő irányban rajzolt, törött vonalból álló mintával lehet nyitni. A minta megadását egy szabályos tízszög segíti, amelynek csúcsait **0**-tól **9**-ig sorszámozták, így a leghosszabb használható minta 10 számjegyet tartalmazhat. Az ajtót nyitó kódszám megadásánál csupán az alakzat és annak iránya érdekes, ezért a **135** mintával nyitható zárat a **802** is nyitja (vagy akár a **024** kódszám is), de a **208** nem. Tehát ebben a mintában a zár csak az óramutató járásával megegyező irányban nyílik. A nyitás az egyes számok egymást követő megérintésével történik.



Az *ajto.txt* fájl soronként egy-egy nyitási próbálkozás adatait tartalmazza. A fájlban legfeljebb 500 sor, soronként legalább 3, legfeljebb 10 karakter lehet.

Készítsen programot, amely az *ajto.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *zar* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el az *ajto.txt* fájl tartalmát!
2. Kérjen be a felhasználótól egy számjegysorozatot, amely a zár kódszáma lesz! (Feltételezheti, hogy a felhasználó ismétlődés nélküli jelsorozatot ad meg.) A teszteléshez használhatja a **239451** sorozatot is.
3. Jelenítse meg a képernyőn, hogy mely kísérleteknél használták a nyitáshoz pontosan az előző feladatban beolvasott kódszámot! A sorok számát egymástól pontosan egy szóközzel válassza el! (A sorok számozását 1-től kezdje!)
4. Adja meg, hogy melyik az első olyan próbálkozás, amely ismétlődő karaktert tartalmaz! Ha nem volt ilyen, írja ki a „nem volt ismétlődő számjegy” üzenetet! (A sorok számozását 1-től kezdje!)
5. Állítson elő egy, a második feladatban beolvasottal egyező hosszúságú, véletlenszerű, ismétlődés nélküli jelsorozatot, majd a mintának megfelelően jelenítse meg a hosszát és az előállított kódszámot!
6. Készítsen függvényt *nyit* néven az alábbi algoritmus alapján, amely a neki átadott két kódszámról megállapítja, hogy ugyanazt a zárat nyitják-e! (A **239451** és a **017239** ugyanazt a zárat nyitja.)
   A függvény két, legfeljebb 10 számjegyből álló karaktersorozathoz egy logikai értéket rendel. A függvény elkészítésekor az algoritmusban megadott változóneveket használja! Az elkészített függvényt a következő feladat megoldásánál felhasználhatja.

```
Függvény nyit(jo, proba:karaktersorozat): logikai érték
   egyezik:=(hossz(jo)=hossz(proba))
   Ha egyezik akkor
       elteres=ascii(jo[1])-ascii(proba[1])
       Ciklus i:=2-től hossz(jo)
          Ha ( elteres - (ascii(jo[i])-ascii(proba[i])) ) mod 10 <> 0
          akkor egyezik:=hamis
       Ciklus vége
   Elágazás vége
   nyit:=egyezik
Függvény vége
```

A mondatszerű leírásban:

- az `a mod b` művelet eredménye az `a` szám `b` számmal történő osztásának maradéka;
- az `ascii()` függvény egy karakterhez annak karakterkódját rendeli.

Az `ascii()` függvény megvalósításához használhatja a következőket az egyes programozási nyelveken:

```
C, C++, C#, Java: (int)karakter; (char)asciikod
Pascal, Python, Perl: ord(karakter); chr(asciikod)
Visual Basic: Asc(karakter); Chr(asciikod)
```

7. Állítsa elő a siker.txt fáljt, amelynek soraiban a nyitási próbálkozás kódszáma után – attól egy szóközzel elválasztva – annak értékelése olvasható.

   - „hibás hossz”, ha a felhasználótól a 2. feladatban bekért kódszám és a sorbeli kódszám hossza eltér;
   - „hibás kódszám”, ha a felhasználótól a 2. feladatban bekért kódszám és a sorbeli kódszám hossza egyezik, de nem összetartozók;
   - „sikeres”, ha a két kódszám egyenértékű.

Minta a szöveges kimenetek kialakításához:

```
2. feladat
Adja meg, mi nyitja a zárat! 239451
3. feladat
A nyitó kódszámok sorai: 1 4 5 8 10…
4. feladat
Az első ismétlődést tartalmazó próbálkozás sorszáma: 9
5. feladat
Egy 6 hosszú kódszám: 078695
```

Részlet a *siker.txt* fájlból:

```
239451 sikeres
154932 hibás kódszám
340562 sikeres
…
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Zár
 *
 * @author Klemand
 */

public class EmeltInfo2016mid {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az ajto.txt fájl beolvasása \n");

		BufferedReader beolvas = new BufferedReader(new FileReader("ajto.txt"));

		String[] ajto = new String[500];
		int db = 0;
		String sor;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			ajto[db - 1] = sor;
		}

		beolvas.close();

		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Egy kódszám bekérése");

		Scanner sc = new Scanner(System.in);
		System.out.print("Adja meg, mi nyitja a zárat (max. 10 karakter, pl. 239451): ");
		String nyitokod = sc.nextLine();
		sc.close();

		System.out.println("\nA 3. feladat megoldása");
		System.out.println("Mikor használták pontosan a megadott kódot?");
		System.out.print("A nyitó kódszámok sorai:");

		int i;
		for (i = 1; i <= db; i++) {
			if (ajto[i - 1].equals(nyitokod)) {
				System.out.print(" " + i);
			}
		}
		System.out.println("\n");
```

```
System.out.println("A 4. feladat megoldása");
System.out.print("Az első ismétlődést tartalmazó próbálkozás sorszáma: ");

boolean[] szerepel = new boolean[10];
boolean ismetles = false;
String aktKod;
int hossz;
int aktSzamjegy;

int j;
i = 0;
while (i < db && !ismetles) {
        i++;
        aktKod = ajto[i - 1];
        hossz = aktKod.length();

        for (j = 0; j <= 9; j++) {
                szerepel[j] = false;
        }

        for (j = 1; j <= hossz; j++) {
                aktSzamjegy = Integer.parseInt(aktKod.substring(j - 1, j));
                if (!szerepel[aktSzamjegy]) {
                        szerepel[aktSzamjegy] = true;
                } else {
                        ismetles = true;
                }
        }
}
if (ismetles) {
        System.out.println(i);
} else {
        System.out.println("nem volt ismétlődő számjegy.");
}

System.out.println("\nAz 5. feladat megoldása");
System.out.println("A nyitókóddal azonos hosszúságú ismétlődés nélküli");
System.out.println("véletlen kódszám generálása");
hossz = nyitokod.length();
System.out.print("Egy " + hossz + " hosszúságú kódszám: ");

int szamjegy;
for (j = 0; j <= 9; j++) {
        szerepel[j] = false;
}

String veletlenKod = "";
do {
        szamjegy = (int) ((Math.random() * 10));
        if (!szerepel[szamjegy]) {
                veletlenKod += szamjegy;
                szerepel[szamjegy] = true;
        }
} while (veletlenKod.length() < hossz);

System.out.println(veletlenKod);

System.out.println("\nA 6. feladat megoldása");
System.out.println("A nyit függvény elkészítése\n");
// A függvény a main metódus után található
```

```
                System.out.println("A 7. feladat megoldása");

                System.out.println("A nyitási próbálkozások eredményének kiírása az siker.txt fájlba. ");

                PrintWriter kiir = new PrintWriter(new FileWriter("siker.txt"));

                for (i = 1; i <= db; i++) {
                        kiir.print(ajto[i - 1]);

                        if (ajto[i - 1].length() != nyitokod.length()) {
                                kiir.println(" hibás hossz");
                        } else {
                                if (nyit(nyitokod, ajto[i - 1])) {
                                        kiir.println(" sikeres");
                                } else {
                                        kiir.println(" hibás kódszám");
                                }
                        }
                }
                kiir.close();
                System.out.println("A fájlkiírás megtörtént. \n");

        }

        public static boolean nyit(String jo, String proba) {
                boolean egyezik = (jo.length() == proba.length());
                if (egyezik) {
                        int elteres = (int) jo.charAt(0) - (int) proba.charAt(0);
                        for (int i = 2; i <= jo.length(); i++) {
                                if ((elteres - ((int) jo.charAt(i - 1) - (int) proba.charAt(i - 1))) % 10 != 0) {
                                        egyezik = false;
                                }
                        }
                }
                return egyezik;
        }

}
```

```
Az 1. feladat megoldása
Az ajto.txt fájl beolvasása

A beolvasás megtörtént.

A 2. feladat megoldása
Egy kódszám bekérése
Adja meg, mi nyitja a zárat (max. 10 karakter, pl. 239451): 239451

A 3. feladat megoldása
Mikor használták pontosan a megadott kódot?
A nyitó kódszámok sorai: 1 4 5 8 10

A 4. feladat megoldása
Az első ismétlődést tartalmazó próbálkozás sorszáma: 9

Az 5. feladat megoldása
A nyitókóddal azonos hosszúságú ismétlődés nélküli
véletlen kódszám generálása
Egy 6 hosszúságú kódszám: 465097

A 6. feladat megoldása
A nyit függvény elkészítése

A 7. feladat megoldása
A nyitási próbálkozások eredményének kiírása az siker.txt fájlba.
A fájlkiírás megtörtént.
```

A siker.txt szövegfájl

```
239451 sikeres
154932 hibás kódszám
340562 sikeres
239451 sikeres
239451 sikeres
451673 sikeres
238451 hibás kódszám
239451 sikeres
2393451 hibás hossz
239451 sikeres
017239 sikeres
673895 sikeres
335422 hibás kódszám
415677 hibás kódszám
```

2016 október: Telefonos ügyfélszolgálat

Egy kis cég ügyfélszolgálata 8 és 12 óra között várja az érdeklődőket. Egyszerre egy hívást tudnak fogadni. A hívások végén azonnal bekapcsolják a következő hívást.

A hívások irányítását egy automata végzi. Nyitáskor és később is – amint a munkatárs szabaddá válik – a legrégebben várakozót kapcsolja be. A munkaidőben érkező hívások esetén – ha a hívónak várnia kell – közli vele a várakozók számát. Munkaidőn kívül érkező hívás esetén az automata a legközelebbi időpontot jelzi az ügyfélnek, aki akár vonalban is maradhat addig. A munkatársnak az összes, a munkaidő vége előtt beérkezett hívást fogadnia kell – tehát a 12:00:00-kor érkezőt már nem –, még akkor is, ha a bekapcsolásukra már a munkaidő befejezése után kerül sor.

A hívások adatait (a kapcsolat létrehozásának és a vonal bontásának időpontját) a *hivas.txt* fájl tárolja a híváskezdés időpontjának sorrendjében. Minden sor két időpontot tartalmaz óra, perc, másodperc formában. A hat számot pontosan egy szóköz választja el egymástól. A sorok száma legfeljebb 1000. Az adatok egy napra vonatkoznak, munkaidőn kívüli értékeket is tartalmazhatnak, minden hívás ezen a napon kezdődött, és be is fejeződött a nap végéig. Feltételezheti, hogy van – legalább két – munkaidőbe eső hívás is. A hívót – a könnyebb kezelhetőség érdekében – a feladatban az időadat sorszámával azonosítjuk.

Például:

```
7 57 36 7 59 59
7 58 5 8 1 39
7 58 33 7 58 47
8 0 1 8 4 17
8 0 21 8 2 13
…
```

A példában egy fájl első 5 sora látható. Ebben az esetben a 2. sor azt mutatja, hogy a hívás a munkaidő kezdete előtt érkezett, de a hívó kivárta, hogy az ügyfélszolgálatos fogadja a hívást. Beszélgetésük 8:0:0-kor kezdődött és 8:1:39-ig tartott, tehát pontosan 99 másodpercig. A 4. hívó megvárta, míg a 2. hívó befejezi, ő 8:1:39-től 8:4:17-ig beszélt az ügyfélszolgálatossal. Az 5. hívóval az automata azt közölte, hogy vele együtt 2 várakozó hívás van. Ő nem várta meg, hogy rá kerüljön a sor.

Látható, hogy egy hívó akkor tudott az ügyfélszolgálatossal beszélni, ha a hívását 12 óra előtt kezdte, valamint 8 óra után, és az összes korábbi hívás végénél később fejezte be.

Készítsen programot, amely a `hivas.txt` állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `telefon` néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Készítse el az `mpbe` függvényt, amely az óra, perc, másodperc `alakban` megadott időpont másodpercben kifejezett értékét adja! A függvényt a megoldásba be kell építenie!

```
Függvény mpbe(o, p, mp:egész szám):egész szám
```

2. Olvassa be a hivas.txt állományban talált adatokat, s annak felhasználásával oldja meg a következő feladatokat!

3. Készítsen statisztikát, amely megadja, hogy óránként hány hívás futott be! A képernyőn soronként egy óra-darabszám párost jelenítsen meg! Csak azok az órák jelenjenek meg, amelyben volt hívás!

4. Írja a képernyőre a leghosszabb hívásnak a sorszámát és másodpercben kifejezett hosszát – attól függetlenül, hogy a hívó tudott-e beszélni az ügyfélszolgálatossal vagy sem! Azonos híváshossz esetén elegendő egyet megjelenítenie.

5. Olvasson be egy munkaidőn belüli időpontot, majd jelenítse meg a képernyőn, hogy hányadik hívóval beszélt akkor az alkalmazott, és éppen hányan vártak arra, hogy sorra kerüljenek! Ha nem volt hívó, akkor a „`Nem volt beszélő.`” üzenetet jelenítse meg!

6. Írja a képernyőre, annak a hívónak az azonosítóját, akivel a munkatárs utoljára beszélt! Írja ki a várakozás másodpercekben mért hosszát is! (Ha nem kellett várnia, a várakozási idő 0.)

7. Készítse el a *`sikeres.txt`* állományt, amely az ügyfélszolgálathoz bekapcsolt hívások listáját tartalmazza! A fájl egyes soraiban a hívó sorszáma, a beszélgetés kezdete (amikor az ügyfélszolgálatos fogadta a hívást) és vége szerepeljen az alábbi mintának megfelelő formában! Például a feladat elején olvasható példa bemenet esetén a fájl tartalma:

```
2 8 0 0 8 1 39
4 8 1 39 8 4 17
…
```

Példa a szöveges kimenetek kialakításához:

```
3. feladat
6 ora 13 hivas
7 ora 89 hivas
…

4. feladat
A leghosszabb ideig vonalban levo hivo 152. sorban szerepel,
a hivas hossza: 341 masodperc.

5. feladat
Adjon meg egy idopontot! (ora perc masodperc) 10 11 12
A varakozok szama: 4 a beszelo a 272. hivo.

6. feladat
Az utolso telefonalo adatai a(z) 432. sorban vannak,
184 masodpercig vart.
```

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Telefonos ügyfélszolgálat
 *
 * @author Klemand
 */

public class EmeltInfo2016okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az mpbe függvény elkészítése\n");
		// A függvény a main metódus után található

		System.out.println("A 2. feladat megoldása");
		System.out.println("A hivas.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("hivas.txt"));

		int[] hivas = new int[1000];
		int[] bontas = new int[1000];
		int db = 0;
		String sor;
		String[] daraboltSor;

		int i;
		while ((sor = beolvas.readLine()) != null) {
			db++;
			daraboltSor = sor.split(" ");
			hivas[db - 1] = mpbe(Integer.parseInt(daraboltSor[0]), Integer.parseInt(daraboltSor[1]),
					Integer.parseInt(daraboltSor[2]));
			bontas[db - 1] = mpbe(Integer.parseInt(daraboltSor[3]), Integer.parseInt(daraboltSor[4]),
					Integer.parseInt(daraboltSor[5]));
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 3. feladat megoldása");
		System.out.println("Az óránkénti hívások száma: ");

		int[] oraDb = new int[24];
		for (i = 0; i <= 23; i++) { // A számlálók nullázása
			oraDb[i] = 0;
		}

		int aktOra;
		for (i = 1; i <= db; i++) {
			aktOra = hivas[i - 1] / 3600;
			oraDb[aktOra]++;
		}

		for (i = 0; i <= 23; i++) {
			if (oraDb[i] > 0) {
				System.out.println(i + " óra " + oraDb[i] + " hívás");
			}
		}
```

```
System.out.println("\nA 4. feladat megoldása");
System.out.println("A leghosszabb hívás adatai");

int max = 1;

for (i = 2; i <= db; i++) {
        if (bontas[i - 1] - hivas[i - 1] > bontas[max - 1] - hivas[max - 1]) {
                max = i;
        }
}

System.out.println("A leghosszabb ideig vonalban levő hívó " + max + ". sorban szerepel,");
System.out.println("a hivás hossza: " + (bontas[max - 1] - hivas[max - 1]) + " másodperc. \n");

System.out.println("Az 5. feladat megoldása");
System.out.println("Egy 8 és 12 óra közötti időpont bekérése");

Scanner sc = new Scanner(System.in);
System.out.print("Adjon meg egy időpontot! (óra perc másodperc): ");
String idopont = sc.nextLine();
daraboltSor = idopont.trim().split(" ");
int ora = Integer.parseInt(daraboltSor[0]);
int perc = Integer.parseInt(daraboltSor[1]);
int mp = Integer.parseInt(daraboltSor[2]);
int idopontMp = mpbe(ora, perc, mp);
int beszelo = -1;
int varakozoDb = 0;
sc.close();

i = 1;
while (i <= db && (bontas[i - 1]) < idopontMp) {
        i++;
}
if (i <= db && (hivas[i - 1] < idopontMp)) {
        beszelo = i;
} else {
        System.out.println("Nem volt beszélő.");
}

if (beszelo > -1) {
        i = beszelo + 1;
        for (i = beszelo + 1; i <= db; i++) {
                if (hivas[i - 1] <= idopontMp && bontas[i - 1] >= idopontMp) {
                        varakozoDb++;
                }
        }
}
System.out.print("Az alkalmazott a(z) " + beszelo + ". hívóval beszélt, ");
System.out.print("a várakozók száma: " + varakozoDb + " volt.\n");

System.out.println("\nA 6. feladat megoldása");

// Az lesz az utolsó telefonáló,
// aki a legkésőbb fejezte be a 12 óra előtti hívók közül
int u = 1;
for (i = 2; i <= db; i++) {
        if (hivas[i - 1] < 12 * 3600 && bontas[i - 1] > bontas[u - 1]) {
                u = i;
        }
}
```

```
                // Az lesz az utolsó előtti telefonáló,
                // aki a legkésőbb fejezte be az utolsó előtti hívók közül
                int ue = 1;
                for (i = 2; i < u; i++) {
                        if (hivas[i - 1] < 12 * 3600 && bontas[i - 1] > bontas[ue - 1]) {
                                ue = i;
                        }
                }

                // A várakozás az utolsó előtti beszélgetés végének
                // és az utolsó hívás kezdetének különbsége vagy 0
                int varakozas = bontas[ue - 1] - hivas[u - 1];
                if (varakozas < 0) {
                        varakozas = 0;
                }
                System.out.print("Az alkalmazottal utoljára beszélő telefonáló sorszáma " + u);
                System.out.println(", a várakozása " + varakozas + " másodpercig tartott.\n");

                System.out.println("A 7. feladat megoldása");
                System.out.println("A sikeres hívások kiírása az sikeres.txt fájlba. ");

                PrintWriter kiir = new PrintWriter(new FileWriter("sikeres.txt"));

                String kapcsolas;
                String befejezes;

                // Az első sikeres hívás megkeresése
                i = 1;
                while (i <= db && (bontas[i - 1] < 8 * 3600)) {
                        i++;
                }
                int e = i; // első hívás
                kapcsolas = idoKonvertalas(Math.max(8 * 3600, hivas[i - 1]));
                int befejezesMp = bontas[e - 1];
                befejezes = idoKonvertalas(befejezesMp);
                kiir.println("" + e + kapcsolas + befejezes);

                // Az utolsó sikeres hívást már ismerjük.
                // Minden új sikeres beszélgetés kezdete az előző befejezésével
                // vagy a saját kezdetével azonos.
                for (i = e + 1; i <= u; i++) {
                        if (bontas[i - 1] > befejezesMp) {
                                kapcsolas = idoKonvertalas(Math.max(befejezesMp, hivas[i - 1]));
                                befejezesMp = bontas[i - 1];
                                befejezes = idoKonvertalas(befejezesMp);
                                kiir.println("" + i + kapcsolas + befejezes);
                        }
                }
                kiir.close();
                System.out.println("A fájlkiírás megtörtént. \n");

        }

        public static int mpbe(int o, int p, int mp) {
                return o * 3600 + p * 60 + mp;
        }

        public static String idoKonvertalas(int ido) {
                int ora, perc, mp;
                mp = ido % 60;
                perc = ido / 60;
                ora = perc / 60;
                perc = perc % 60;
                return " " + ora + " " + perc + " " + mp;
        }

}
```

```
Az 1. feladat megoldása
Az mpbe függvény elkészítése

A 2. feladat megoldása
A hivas.txt fájl beolvasása
A beolvasás megtörtént.

A 3. feladat megoldása
Az óránkénti hívások száma:
6 óra 13 hívás
7 óra 89 hívás
8 óra 78 hívás
9 óra 80 hívás
10 óra 85 hívás
11 óra 88 hívás
12 óra 18 hívás
13 óra 2 hívás
15 óra 2 hívás
17 óra 1 hívás
18 óra 2 hívás

A 4. feladat megoldása
A leghosszabb hívás adatai
A leghosszabb ideig vonalban levő hívó 152. sorban szerepel,
a hivás hossza: 341 másodperc.

Az 5. feladat megoldása
Egy 8 és 12 óra közötti időpont bekérése
Adjon meg egy időpontot! (óra perc másodperc): 10 11 12
Az alkalmazott a(z) 272. hívóval beszélt, a várakozók száma: 4 volt.

A 6. feladat megoldása
Az alkalmazottal utoljára beszélő telefonáló sorszáma 432, a várakozása 184 másodpercig tartott.

A 7. feladat megoldása
A sikeres hívások kiírása az sikeres.txt fájlba.
A fájlkiírás megtörtént.
```

A sikeres.txt szövegfájl

```
96 8 0 0 8 0 30
98 8 0 30 8 1 36
100 8 1 36 8 1 39
102 8 1 39 8 4 17
103 8 4 17 8 5 56
104 8 5 56 8 6 37
...
200 9 13 38 9 14 44
201 9 15 26 9 19 11
206 9 19 11 9 20 39
208 9 20 39 9 23 31
209 9 23 31 9 24 48
214 9 24 48 9 26 28
...
421 11 55 37 11 57 10
422 11 57 10 11 57 43
423 11 57 43 11 58 37
425 11 58 37 12 0 22
428 12 0 22 12 2 29
432 12 2 29 12 3 46
```

2017 május: Tesztverseny

Egy közismereti versenyen a versenyzőknek 13+1, azaz összesen 14 tesztfeladatot tűznek ki. A versenyzőknek minden feladat esetén négy megadott lehetőség (A, B, C, D) közül kell a helyes választ megjelölniük. A versenybizottság garantálja, hogy tesztlapon minden kérdéshez pontosan egy helyes válasz tartozik. A kitöltött tesztlapokat elektronikusan rögzítik, a visszaélések elkerülése végett a versenyzőket betűkből és számokból álló kóddal azonosítják.

A helyes megoldást és a versenyzők válaszait a *valaszok.txt* szöveges állomány tartalmazza. A fájlban legfeljebb 500 versenyző adatai szerepelnek. A fájl első sorában a helyes válaszok szerepelnek. A fájl többi sora a versenyzők kódjával kezdődik, ezt egy szóköz, majd az adott versenyző által adott válaszok sorozata követi. A versenyzők kódja legfeljebb 5 karakterből áll. A válaszok a feladatokkal egyező sorrendben, elválasztójel nélkül, nagybetűvel szerepelnek. Ha a versenyző egy kérdésre nem válaszolt, akkor annak helyén X betű szerepel. Például:

```
BCCCDBBBBCDAAA
AB123 BXCDBBACACADBC
AH97 BCACDBDDBCBBCA
…
```

A 2. kérdésre a helyes válasz a C volt, de erre a kérdésre az AB123 kódú versenyző nem válaszolt.

Készítsen programot *tesztverseny* néven az alábbi feladatok megoldására! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `2. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! A képernyőn megjelenő üzenetek az adott környezet nyelvi sajátosságainak megfelelően a mintától eltérhetnek (pl. ékezetmentes betűk, tizedespont használata).

1. Olvassa be és tárolja el a *valaszok.txt* szöveges állomány adatait!
2. Jelenítse meg a képernyőn a mintának megfelelően, hogy hány versenyző vett részt a tesztversenyen!
3. Kérje be egy versenyző azonosítóját, és jelenítse meg a mintának megfelelően a hozzá eltárolt válaszokat! Feltételezheti, hogy a fájlban létező azonosítót adnak meg.
4. Írassa ki a képernyőre a helyes megoldást! A helyes megoldás alatti sorba „+” jelet tegyen, ha az adott feladatot az előző feladatban kiválasztott versenyző eltalálta, egyébként egy szóközt! A kiírást a mintának megfelelő módon alakítsa ki!
5. Kérje be egy feladat sorszámát, majd határozza meg, hogy hány versenyző adott a feladatra helyes megoldást, és ez a versenyzők hány százaléka! A százalékos eredményt a mintának megfelelően, két tizedesjeggyel írassa ki!
6. A verseny feladatai nem egyenlő nehézségűek: az 1-5. feladat 3 pontot, a 6-10. feladat 4 pontot, a 11-13. feladat 5 pontot, míg a 14. feladat 6 pontot ér. Határozza meg az egyes versenyzők pontszámát, és a listát írassa ki a *pontok.txt* nevű állományba! Az állomány minden sora egy versenyző kódját, majd szóközzel elválasztva az általa elért pontszámot tartalmazza!
7. A versenyen a három legmagasabb pontszámot elérő összes versenyzőt díjazzák. Például 5 indulónál előfordulhat, hogy 3 első és 2 második díjat adnak ki. Így megtörténhet az is, hogy nem

kerül sor mindegyik díj kiadására. Írassa ki a mintának megfelelően a képernyőre a díjazottak kódját és pontszámát pontszám szerint csökkenő sorrendben!

Minta a szöveges kimenetek kialakításához:

(A képernyőre írt üzeneteknek tartalmilag meg kell felelniük az alábbi mintának. Képernyőre írást nem igénylő feladatok esetén nem szükséges a feladat számát sem kiíratnia.)

```
1. feladat: Az adatok beolvasása

2. feladat: A vetélkedőn 303 versenyző indult.

3. feladat: A versenyző azonosítója = AB123
BXCDBBACACADBC   (a versenyző válasza)

4. feladat:
BCCCDBBBBCDAAA   (a helyes megoldás)
+ +  +   +       (a versenyző helyes válaszai)

5. feladat: A feladat sorszáma = 10
A feladatra 111 fő, a versenyzők 36,63%-a adott helyes választ.

6. feladat: A versenyzők pontszámának meghatározása

7. feladat: A verseny legjobbjai:
1. díj (56 pont): JO001
2. díj (52 pont): DG490
2. díj (52 pont): UA889
3. díj (49 pont): FX387
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Tesztverseny
 *
 * @author Klemand
 */

public class EmeltInfo2017maj {

        public static void main(String[] args) throws IOException {

                System.out.println("Az 1. feladat megoldása");
                System.out.println("A valaszok.txt fájl beolvasása");

                BufferedReader beolvas = new BufferedReader(new FileReader("valaszok.txt"));

                String megoldas = beolvas.readLine();

                String[] versenyzo = new String[500];
                String[] valasz = new String[500];
                int db = 0;
                String sor;
                String[] daraboltSor;

                while ((sor = beolvas.readLine()) != null) {
                        db++;
                        daraboltSor = sor.split(" ");
                        versenyzo[db - 1] = daraboltSor[0];
                        valasz[db - 1] = daraboltSor[1];
                }
                beolvas.close();
                System.out.println("A beolvasás megtörtént.\n");

                System.out.println("A 2. feladat megoldása");
                System.out.println("A vetélkedőn " + db + " versenyző indult.\n");

                System.out.println("A 3. feladat megoldása");
                System.out.print("Adja meg egy versenyző azonosítóját:  ");

                Scanner sc1 = new Scanner(System.in);
                String aktVersenyzo = sc1.nextLine();

                int i = 1;
                while (i <= db && versenyzo[i].equals(aktVersenyzo)) {
                        i++;
                }
                if (i <= db) {
                        System.out.println(valasz[i - 1] + "\t (a versenyző válasza) \n");
                } else {
                        System.out.println("Nincs ilyen versenyző.");
                }
```

```
System.out.println("A 4. feladat megoldása");

System.out.println(megoldas + "\t (a helyes megoldás)");
String aktValasz = valasz[i-1];
int j;
for (j = 1; j <= megoldas.length(); j++) {
        if (aktValasz.substring(j - 1, j).equals(megoldas.substring(j - 1, j))) {
                System.out.print("+");
        } else {
                System.out.print(" ");
        }
}
System.out.println("\t (a versenyző helyes válaszai)");

System.out.println("\nAz 5. feladat megoldása");
System.out.print("Kérem egy feladat sorszámát (max 14): ");

Scanner sc2 = new Scanner(System.in);
int sorszam = sc1.nextInt();
int joDb = 0;
for (i = 1; i <= db; i++) {
        if (valasz[i - 1].substring(sorszam - 1, sorszam).equals(megoldas.substring(sorszam - 1, sorszam))) {
                joDb++;
        }
}
System.out.print("A feladatra " + joDb + " fő, a versenyzők ");
System.out.printf("%.2f", ((double) joDb / db) * 100);
System.out.println("%-a adott helyes választ.\n");
sc1.close();
sc2.close();

System.out.println("A 6. feladat megoldása");

System.out.println("A versenyzők pontszámainak kiírása a pontok.txt fájlba.");

// Tároljuk a pontszámokat, mert a következő feladatban is szükség lesz rájuk.
int[] pontok = new int[500];

PrintWriter kiir = new PrintWriter(new FileWriter("pontok.txt"));
for (i = 1; i <= db; i++) {
        pontok[i - 1] = pontMeghat(valasz[i - 1], megoldas);
        kiir.println(versenyzo[i - 1] + " " + pontok[i - 1]);
}
kiir.close();
System.out.println("A fájlkiírás megtörtént. \n");
```

```
                System.out.println("A 7. feladat megoldása");
                System.out.println("A verseny legjobbjai: ");
                // A versenyzők rendezése pontszámok szerint csökkenő sorrendbe
                // A válaszokra most nincs szükségünk, de azokat is bevonjuk a rendezésbe.

                String stringAsztal;
                int intAsztal;

                for (i = db - 1; i >= 1; i--) {
                        for (j = 1; j <= i; j++) {
                                if (pontok[j - 1] < pontok[j]) {
                                        stringAsztal = versenyzo[j - 1];
                                        versenyzo[j - 1] = versenyzo[j];
                                        versenyzo[j] = stringAsztal;

                                        stringAsztal = valasz[j - 1];
                                        valasz[j - 1] = valasz[j];
                                        valasz[j] = stringAsztal;

                                        intAsztal = pontok[j - 1];
                                        pontok[j - 1] = pontok[j];
                                        pontok[j] = intAsztal;
                                }
                        }
                }

                int dij = 1;
                int aktPont = pontok[0];
                i = 1;
                do {
                        System.out.println(dij + ". díj (" + pontok[i - 1] + " pont): " + versenyzo[i - 1]);
                        i++;
                        if (pontok[i - 1] < aktPont) {
                                aktPont = pontok[i - 1];
                                dij++;
                        }
                } while (i <= db && dij <= 3);

        }

        public static int pontMeghat(String valasz, String megoldas) {
                int s;
                int pont = 0;
                for (s = 1; s <= megoldas.length(); s++) {
                        if (valasz.substring(s - 1, s).equals(megoldas.substring(s - 1, s))) {
                                if (s == 14) {
                                        pont += 6;
                                } else if (s >= 11) {
                                        pont += 5;
                                } else if (s >= 6) {
                                        pont += 4;
                                } else {
                                        pont += 3;
                                }
                        }
                }
                return pont;
        }

}
```

```
Az 1. feladat megoldása
A valaszok.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
A vetélkedőn 303 versenyző indult.

A 3. feladat megoldása
Adja meg egy versenyző azonosítóját:  AB123
BXCDBBACACADBC    (a versenyző válasza)

A 4. feladat megoldása
BCCCDBBBBCDAAA    (a helyes megoldás)
+ +  +   +        (a versenyző helyes válaszai)

Az 5. feladat megoldása
Kérem egy feladat sorszámát (max 14): 10
A feladatra 111 fő, a versenyzők 36,63%-a adott helyes választ.

A 6. feladat megoldása
A versenyzők pontszámainak kiírása a pontok.txt fájlba.
A fájlkiírás megtörtént.

A 7. feladat megoldása
A verseny legjobbjai:
1. díj (56 pont): JO001
2. díj (52 pont): DG490
2. díj (52 pont): UA889
3. díj (49 pont): FX387
```

A pontok.txt szövegfájl

```
AB123 14
AD995 27
AH97 30
AK260 15
AL580 32
AN562 7
AN784 20
AQ258 22
BC476 4
BC504 10
BF417 4
BF811 3
BG792 0
BK550 4
BL262 17
BM577 0
BP156 19
BT532 4
BV584 16
CI121 9
CI210 31
CK500 8
CK895 20
CQ999 33
CU274 20
CU716 29
CV554 23
CX616 16
CX617 19
CY610 31
CY823 9
DG490 52
DH495 6
DI788 33
DM396 11
DR214 8
DS234 7
DS847 18
DV112 31
DW557 20
DW652 18
EA690 0
EC330 9
EC465 6
ED435 16
EF827 34
EG445 14
EM426 4
EO997 4
EP258 21
ES386 18
ES737 38
ET473 31
EU253 8
EZ790 17
FB231 9
FD551 38
FF166 17
FJ530 25
FJ576 17
FM114 7
FM712 4
FQ313 10
FR961 8
FV130 24
FX215 28
FX387 49
FX698 34
GA563 22
GG434 23
GG624 32
GJ813 8
GL574 13
GL716 21
GN159 31
GQ239 32
GS212 33
GS249 19
GW599 18
GX667 14
GX860 19
HE378 6
HG142 29
HH263 17
HJ949 23
HK731 19
HL460 7
HQ253 44
HQ443 17
HS886 8
HT166 34
HX170 15
HY470 4
HZ276 9
ID551 21
IE789 19
IJ423 32
IT262 7
IT764 32
IU238 31
IW569 41
JB651 21
JB831 32
JC124 19
JD329 31
JD830 31
JI428 32
JL132 17
JL404 16
JO001 56
JP806 13
JX172 34
JY151 28
JY212 4
KB191 35
KB217 3
KD869 11
KH563 14
KI171 21
KN731 17
KP396 4
KP542 17
KQ775 21
KX675 20
KX870 3
KY681 10
KZ132 33
...
```

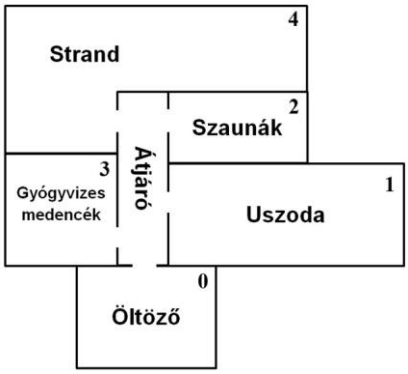
2017 május idegennyelvű: Fürdő

A fürdőkben egyre gyakoribb a különböző beléptető és fürdőn belüli mozgást rögzítő rendszerek alkalmazása. Egy fürdő a szolgáltatások fejlesztése miatt szeretné a vendégek fürdőzési szokásait felmérni. Ezért egy napi forgalomból véletlenszerűen választották ki a vendégek adatait.

A fürdő négy elkülönített részleggel rendelkezik. A vendégek a fürdő részlegeit az öltözőből kilépve az átjárón keresztül érhetik el, és a fürdőből távozni is az öltözőn keresztül tudnak. Minden vendég a belépéskor egy karszalagot kap. A karszalagon lévő érzékelő minden részlegbe való belépést és kilépést rögzít. Minden vendég az öltözőt egyszer hagyja el – ekkor lép a fürdő belső területére –, és egyszer megy be az öltözőbe – ekkor hagyja el a fürdőt. A nap folyamán már nem jön vissza ismét a fürdőbe. A fürdő 6 órától 20 óráig tart nyitva.

A szóközökkel tagolt *furdoadat.txt* fájl maximálisan 800 adatsort tartalmazhat. A fájlban 100 fürdővendég adatai vannak. A lista vendégenként csoportosított, azon belül idő szerint rendezett. A vendégek sorrendjét az öltözőből való kilépés ideje szabja meg.

Részleg	Azonosító
Öltöző	0
Uszoda	1
Szaunák	2
Gyógyvizes medencék	3
Strand	4



- A sor első értéke egy háromjegyű szám, ami a vendég azonosítója.
- A sor második értéke a fürdőrészleg azonosítója.
- A sor harmadik értéke 0, ha a vendég az adott részlegre belépett; és 1, ha kilépett a részlegből.
- A sor negyedik, ötödik és hatodik értéke az adott részlegbe való belépés vagy kilépés időpontja óra perc másodperc formában, 24 órás alakban.

Például:

```
453 0 1 6 15 27
453 1 0 6 17 19
453 1 1 6 52 56
453 0 0 6 56 32
…
266 0 1 16 7 52
266 4 0 16 9 30
…
```

A példában a 453-as és a 266-os azonosítóval rendelkező vendég néhány adata látható. A 453-as vendég 6:15:27-kor lépett ki az öltözőből és 6:17:19-kor lépett be az uszodába.
Az uszodából 6:52:56-kor lépett ki, majd 6:56:32-kor bement az öltözőbe.

Készítsen programot, amely a *furdoadat.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *furdostat* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `4. feladat`)! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be a *furdoadat.txt* fájl tartalmát!
2. Írja a képernyőre, hogy az első és az utolsó vendég mikor lépett ki az öltözőből!
3. Határozza meg és írja ki a képernyőre, hogy hány olyan fürdővendég volt, aki az öltözőn kívül csak egy részlegen járt és azt a részleget csak egyszer használta!
4. Határozza meg, hogy melyik vendég töltötte a legtöbb időt a fürdőben! A vendég azonosítóját és a fürdőben tartózkodás idejét írja ki a képernyőre! A fürdőben a legtöbb időt töltő vendégek közül elegendő egy vendég adatait megjelenítenie.
5. Készítsen statisztikát, hogy 06:00:00-08:59:59 óra között, 09:00:00-15:59:59 óra között és 16:00:00-19:59:59 óra között hány vendég érkezett a fürdőbe! Az eredményt írja ki a képernyőre a mintán látható formában!
6. Készítsen egy listát a szauna részlegen járt vendégekről és az általuk ott töltött időről! A vendég azonosítóját és a részlegen eltöltött időt a *szauna.txt* fájlba írja ki! A fájlban egy sorban a vendég azonosítója és szóközzel elválasztva a részlegen eltöltött idő szerepeljen óra:perc:másodperc formában! Ügyeljen arra, hogy egy vendég a szauna részlegben a nap folyamán többször is járhatott!
7. Készítsen egy listát, amelyben megadja, hogy az egyes részlegeket hányan használták! Az eredményt a minta szerint írja ki a képernyőre! Ha egy vendég egy részlegen többször is járt a nap folyamán, azt a statisztikában csak egynek számolja!

Minta a szöveges kimenetek kialakításához:

```
2. feladat
Az első vendég 6:14:56-kor lépett ki az öltözőből.
Az utolsó vendég 18:35:37-kor lépett ki az öltözőből.

3. feladat
A fürdőben 33 vendég járt csak egy részlegen.

4. feladat
A legtöbb időt eltöltő vendég:
306. vendég 6:41:19

5. feladat
6-9 óra között 9 vendég
9-16 óra között 45 vendég
16-20 óra között 46 vendég

7. feladat
Uszoda: 41
Szaunák: 52
Gyógyvizes medencék: 54
Strand: 48
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * Fürdő
 *
 * @author Klemand
 */

public class EmeltInfo2017mid {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A furdoadat.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("furdoadat.txt"));

		int[][] mozgas = new int[800][6];
		// vendég, részleg, ki/be (1/0), óra, perc, mp
		int db = 0;
		String sor;
		String[] daraboltSor;
		int j;
		while ((sor = beolvas.readLine()) != null) {
			db++;
			daraboltSor = sor.split(" ");
			for (j = 1; j <= 6; j++) {
				mozgas[db - 1][j - 1] = Integer.parseInt(daraboltSor[j - 1]);
			}
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.print("Az első vendég " + mozgas[0][3] + ":" + mozgas[0][4] + ":" + mozgas[0][5]);
		System.out.println("-kor lépett ki az öltözőből.");
		int i = db;
		// Kiválasztás, nem szükséges az i >= 1 feltétel!
		while (!(mozgas[i - 1][1] == 0 && mozgas[i - 1][2] == 1)) {
			i--;
		}
		System.out.print("Az utolsó vendég " + mozgas[i - 1][3] + ":" + mozgas[i - 1][4] + ":" + mozgas[i - 1][5]);
		System.out.println("-kor lépett ki az öltözőből.");

		System.out.println("\nA 3. feladat megoldása");

		int egyDb = 0;
		int aktDb = 0;
		for (i = 1; i <= db; i++) {
			if (mozgas[i - 1][1] == 0 && mozgas[i - 1][2] == 1) {
				aktDb = 0;
			}
			aktDb++;
			if (mozgas[i - 1][1] == 0 && mozgas[i - 1][2] == 0 && aktDb == 4) {
				egyDb++;
			}
		}
		System.out.println("A fürdőben " + egyDb + " vendég járt csak egy részlegben.\n");
```

```
System.out.println("A 4. feladat megoldása");
System.out.println("A legtöbb időt eltöltő vendég:");

int maxVendeg = -1;
int maxIdo = 0;
int aktVendeg = -1;
int aktIdo = 0;
for (i = 1; i <= db; i++) {
        if (mozgas[i - 1][1] == 0 && mozgas[i - 1][2] == 1) { // érkezik
                aktVendeg = mozgas[i - 1][0];
                aktIdo = -(mozgas[i - 1][3] * 3600 + mozgas[i - 1][4] * 60 + mozgas[i - 1][5]);
        }
        if (mozgas[i - 1][1] == 0 && mozgas[i - 1][2] == 0) { // távozik
                aktIdo += (mozgas[i - 1][3] * 3600 + mozgas[i - 1][4] * 60 + mozgas[i - 1][5]);
                if (aktIdo > maxIdo) {
                        maxIdo = aktIdo;
                        maxVendeg = aktVendeg;
                }
        }
}
System.out.println(maxVendeg + ". vendég, tartózkodási ideje " + idoKonvertalas(maxIdo));

System.out.println("\nAz 5. feladat megoldása");
System.out.println("A vendégek érkezésének eloszlása");
int estKezdet = 16 * 3600;
int nappalKezdet = 9 * 3600;
int dbR = 0;
int dbN = 0;
int dbE = 0;

for (i = 1; i <= db; i++) {
        if (mozgas[i - 1][1] == 0 && mozgas[i - 1][2] == 1) { // érkezik
                aktIdo = mozgas[i - 1][3] * 3600 + mozgas[i - 1][4] * 60 + mozgas[i - 1][5];
                if (aktIdo >= estKezdet) {
                        dbE++;
                } else if (aktIdo >= nappalKezdet) {
                        dbN++;
                } else {
                        dbR++;
                }
        }
}
System.out.println("6-9 óra között " + dbR + " vendég");
System.out.println("9-16 óra között " + dbN + " vendég");
System.out.println("16-20 óra között " + dbE + " vendég\n");
```

```
System.out.println("A 6. feladat megoldása");

System.out.println("A szauna részleg vendégeinek és a szaunában töltött");
System.out.println("összidejüknek kiírása a szauna.txt fájlba.");

int[] szaunaLista = new int[100];
int szaunaListaDb = 0;
int[] szaunaIdo = new int[100];
aktVendeg = -1;
aktIdo = 0;

for (i = 1; i <= db; i++) {
        if (mozgas[i - 1][1] == 2 && mozgas[i - 1][2] == 0) { // belép a szaunába
                aktVendeg = mozgas[i - 1][0];
                aktIdo = -(mozgas[i - 1][3] * 3600 + mozgas[i - 1][4] * 60 + mozgas[i - 1][5]);
        }
        if (mozgas[i - 1][1] == 2 && mozgas[i - 1][2] == 1) { // kilép a szaunából
                aktIdo += (mozgas[i - 1][3] * 3600 + mozgas[i - 1][4] * 60 + mozgas[i - 1][5]);
                // Ha szerepelt már a szaunalistában, hozzáadjuk az idejét,
                // ha még nem, akkor felvesszük a listába
                j = 1;
                while (j <= szaunaListaDb && !(szaunaLista[j - 1] == aktVendeg)) {
                        j++;
                }
                if (j <= szaunaListaDb) {
                        szaunaIdo[j - 1] += aktIdo;
                } else {
                        szaunaListaDb++;
                        szaunaLista[szaunaListaDb - 1] = aktVendeg;
                        szaunaIdo[szaunaListaDb - 1] = aktIdo;
                }
        }
}

PrintWriter kiir = new PrintWriter(new FileWriter("szauna.txt"));

for (i = 1; i <= szaunaListaDb; i++) {
        kiir.println(szaunaLista[i - 1] + " " + idoKonvertalas(szaunaIdo[i - 1]));
}
kiir.close();
System.out.println("A fájlkiírás megtörtént.");
System.out.println("A szaunának összesen " + szaunaListaDb + " látogatója volt.\n");
```

```
                System.out.println("A 7. feladat megoldása");
                System.out.println("Az egyes részlegek látogatottsága");
                int[][] reszlegLista = new int[100][4];
                int[] reszlegListaDb = { 0, 0, 0, 0 };
                int r;
                for (r = 1; r <= 4; r++) {
                        for (i = 1; i <= db; i++) {
                                if (mozgas[i - 1][1] == r && mozgas[i - 1][2] == 0) { // belép a r-edik részlegbe
                                        aktVendeg = mozgas[i - 1][0];
                                }
                                if (mozgas[i - 1][1] == r && mozgas[i - 1][2] == 1) { // kilép a r-edik részlegből
                                        j = 1;
                                        while (j <= reszlegListaDb[r - 1] && !(reszlegLista[j - 1][r - 1] == aktVendeg)) {
                                                j++;
                                        }
                                        if (j > reszlegListaDb[r - 1]) {
                                                reszlegListaDb[r - 1]++;
                                                reszlegLista[reszlegListaDb[r - 1] - 1][r - 1] = aktVendeg;
                                        }
                                }
                        }
                }

                System.out.println("Uszoda: " + reszlegListaDb[0]);
                System.out.println("Szaunák: " + reszlegListaDb[1]);
                System.out.println("Gyógyvizes medencék: " + reszlegListaDb[2]);
                System.out.println("Strand: " + reszlegListaDb[3]);
                System.out.println();

        }

        public static String idoKonvertalas(int ido) {
                int ora, perc, mp;
                mp = ido % 60;
                perc = ido / 60;
                ora = perc / 60;
                perc = perc % 60;
                return "" + ora + ":" + perc + ":" + mp;
        }

}
```

```
Az 1. feladat megoldása
A furdoadat.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
Az első vendég 6:14:56-kor lépett ki az öltözőből.
Az utolsó vendég 18:35:37-kor lépett ki az öltözőből.

A 3. feladat megoldása
A fürdőben 33 vendég járt csak egy részlegben.

A 4. feladat megoldása
A legtöbb időt eltöltő vendég:
306. vendég, tartózkodási ideje 6:41:19

Az 5. feladat megoldása
A vendégek érkezésének eloszlása
6-9 óra között 9 vendég
9-16 óra között 45 vendég
16-20 óra között 46 vendég

A 6. feladat megoldása
A szauna részleg vendégeinek és a szaunában töltött
összidejüknek kiírása a szauna.txt fájlba.
A fájlkiírás megtörtént.
A szaunának összesen 52 látogatója volt.

A 7. feladat megoldása
Az egyes részlegek látogatottsága
Uszoda: 41
Szaunák: 52
Gyógyvizes medencék: 54
Strand: 48
```

A szauna.txt szövegfájl

```
317 0:14:33
332 0:9:55
421 0:8:4
435 0:12:41
217 0:5:32
130 0:14:16
459 0:13:13
480 0:13:9
289 0:13:33
422 0:13:6
306 0:7:41
148 0:13:32
368 0:10:9
114 0:6:29
237 0:7:30
401 0:6:22
232 0:11:0
377 0:13:21
443 0:14:34
392 0:6:25
171 0:9:31
147 0:9:13
206 0:22:34
191 0:10:27
210 0:10:31
469 0:26:4
354 0:11:31
230 0:14:21
471 0:12:31
254 0:10:30
205 0:14:26
294 0:11:41
133 0:26:40
416 0:13:29
288 0:23:23
486 0:20:37
287 0:14:41
117 0:11:56
425 0:14:38
257 0:11:15
204 0:11:7
389 0:11:44
451 0:24:17
330 0:23:36
131 0:13:54
110 0:12:21
188 0:12:46
428 0:10:47
484 0:22:53
186 0:13:44
224 0:23:41
101 0:14:10
```

2017 október: Hiányzások

Egy osztály második félévi hiányzásai állnak rendelkezésére a *naplo.txt* fájlban. A hiányzások naponként csoportosítva szerepelnek, minden napot a # karakter kezd, majd egyegy szóközzel elválasztva a hónap és a nap sorszáma következik. Az aznapi hiányzások tanulónként külön sorokban vannak, a tanuló napi hiányzásait egy hét karakterből álló karaktersorozat írja le. A karaktersorozat minden karaktere egy-egy órát ad meg. Értéke az O betű, ha a tanuló jelen volt az adott órán, az X utal az igazolt, az I az igazolatlan távollétre, végül N betű jelzi, ha a tanulónak akkor nem volt órája. Például:

```
# 01 15
Galagonya Alfonz OXXXXXN
# 01 16
Alma Hedvig OOOOOIO
Galagonya Alfonz XXXXXXX
```

A fenti példa a január 15-16-i hiányzásokat tartalmazza. Galagonya Alfonznak január 15-én hat órája lett volna, de csak az első órán volt jelen, utána igazoltan hiányzott. Alma Hedvignek január 16-án hét órája lett volna, de a 6. óráról igazolatlanul távol maradt.

Az állomány legfeljebb 600 sort tartalmaz, az osztályba pedig legfeljebb 50 tanuló jár. Feltételezheti, hogy az osztályban nincs két azonos nevű tanuló, továbbá hogy minden tanulónak egy vezeték és egy utóneve van. Felhasználhatja, hogy a jelenlétre vonatkozó bejegyzés mindig 7 karakterből áll.

Készítsen programot, amely az állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját *hianyzasok* néven mentse! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, és feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az eredmények kiírásánál utaljon a kiírt adat jelentésére! A mintától eltérő, valamint az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el a *naplo.txt* fájl tartalmát!
2. Határozza meg és írassa ki, hogy hány sor van a fájlban, ami hiányzást rögzít! (A fenti példában 3 ilyen bejegyzés van.)
3. Számolja meg és írassa ki, hogy összesen hány óra igazolt és hány óra igazolatlan hiányzás volt a félév során!

Néhány tanár azt feltételezi, hogy a tanulók bizonyos órákról gyakrabban hiányoznak. A következő három feladatban ennek vizsgálatát kell előkészítenie.

4. Készítsen függvényt **hetnapja** néven, amely a paraméterként megadott dátumhoz (hónap, nap) megadja, hogy az a hét melyik napjára esik (hétfő, kedd…). Tudjuk, hogy az adott év nem volt szökőév, továbbá azt is, hogy január elseje hétfőre esett. Használhatja az alábbi algoritmust is, ahol a tömbök indexelése 0-val kezdődik, de ettől eltérő megoldású függvényt is készíthet.

```
Függvény hetnapja(honap:egesz, nap:egesz): szöveg
   napnev[]:= ("vasarnap", "hetfo", "kedd", "szerda", "csutortok",
               "pentek", "szombat")
   napszam[]:= (0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 335)
   napsorszam:= (napszam[honap-1]+nap) MOD 7
   hetnapja:= napnev[napsorszam]
Függvény vége
```

5. Kérjen be egy dátumot (hónap, nap), és a **hetnapja** függvény felhasználásával írassa ki, hogy az a hét melyik napjára esett!
6. Kérje be a hét egy tanítási napjának nevét és egy aznapi tanítási óra óraszámát (például: kedd 3)! Írassa ki a képernyőre, hogy a félév során az adott tanítási órára összesen hány hiányzás jutott!
7. Írassa ki a képernyőre a legtöbb órát hiányzó tanuló nevét! Ha több ilyen tanuló is van, akkor valamennyi neve jelenjen meg szóközzel elválasztva!

Minta a szöveges kimenetek kialakításához:

```
2. feladat
A naplóban 139 bejegyzés van.
3. feladat
Az igazolt hiányzások száma 788, az igazolatlanoké 18 óra.
5. feladat
A hónap sorszáma=2
A nap sorszáma=3
Azon a napon szombat volt.
6. feladat
A nap neve=szerda
Az óra sorszáma=3
Ekkor összesen 49 óra hiányzás történt.
7. feladat
A legtöbbet hiányzó tanulók: Kivi Adrienn Jujuba Ibolya
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

/**
 * Hiányzások
 *
 * @author Klemand
 */

public class EmeltInfo2017okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A naplo.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("naplo.txt"));

		String[] nev = new String[600];
		int[] ho = new int[600];
		int[] nap = new int[600];
		String[] hianyzas = new String[600];
		int aktHo = -1;
		int aktNap = -1;
		int db = 0;

		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			daraboltSor = sor.split(" ");
			if (daraboltSor[0].equals("#")) {
				//A # karaktert nem írta ki az Eclipse, a jegyzettömbből másoltam be.
				aktHo = Integer.parseInt(daraboltSor[1]);
				aktNap = Integer.parseInt(daraboltSor[2]);
			} else {
				db++;
				ho[db - 1] = aktHo;
				nap[db - 1] = aktNap;
				nev[db - 1] = daraboltSor[0] + " " + daraboltSor[1];
				hianyzas[db - 1] = daraboltSor[2];
			}
		}
		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("A naplóban " + db + " bejegyzés van.\n");
```

```
System.out.println("A 3. feladat megoldása");

int igDb = 0;
int iglanDb = 0;
String aktBejegyzes;
int i, j;
for (i = 1; i <= db; i++) {
        for (j = 1; j <= 7; j++) {
                aktBejegyzes = hianyzas[i - 1].substring(j - 1, j);
                if (aktBejegyzes.equals("X")) {
                        igDb++;
                }
                if (aktBejegyzes.equals("I")) {
                        iglanDb++;
                }
        }
}
System.out.println("Az igazolt hiányzások száma " + igDb + ",  az igazolatlanoké " + iglanDb + " óra.\n");

System.out.println("A 4. feladat megoldása");

System.out.println("A hetnapja függvény elkészítése\n");
// A függvény a main metódus után található.

System.out.println("Az 5. feladat megoldása");

System.out.print("Kérem egy hónap sorszámát (1 - 12): ");
Scanner sc1 = new Scanner(System.in);
aktHo = sc1.nextInt();
System.out.print("Kérem a hónap napjának sorszámát (1 - 28/30/31): ");
aktNap = sc1.nextInt();
System.out.println("Azon a napon " + hetnapja(aktHo, aktNap) + " volt.\n");

System.out.println("A 6. feladat megoldása");

System.out.print("Kérem a hét egy napjának nevét (ékezetek nélkül): ");

Scanner sc2 = new Scanner(System.in);
String aktNapNev = sc2.nextLine();
System.out.print("Kérem a tanítási óra sorszámát (1 - 7): ");
int aktOra = sc1.nextInt();
int aktHianyzas = 0;

for (i = 1; i <= db; i++) {
        if (hetnapja(ho[i - 1], nap[i - 1]).equals(aktNapNev)) {
                aktBejegyzes = hianyzas[i - 1].substring(aktOra - 1, aktOra);
                if (aktBejegyzes.equals("X") || aktBejegyzes.equals("I")) {
                        aktHianyzas++;
                }
        }
}
System.out.println("Az adott tanítási órára összesen " + aktHianyzas + " hiányzás jutott.\n");
sc1.close();
sc2.close();
```

```
                System.out.println("A 7. feladat megoldása");
                System.out.print("A legtöbbet hiányzó tanulók: ");

                // Statisztika készítése a tanulók hiányzásairól

                String[] tanuloLista = new String[50];
                int tanuloListaDb = 0;
                int[] hianyzasDb = new int[50];
                String aktNev;
                for (i = 1; i <= db; i++) {
                        aktNev = nev[i - 1];
                        aktHianyzas = 0;
                        for (j = 1; j <= 7; j++) {
                                aktBejegyzes = hianyzas[i - 1].substring(j - 1, j);
                                if (aktBejegyzes.equals("X") || aktBejegyzes.equals("I")) {
                                        aktHianyzas++;
                                }
                        }
                        j = 1;
                        while (j <= tanuloListaDb && !tanuloLista[j - 1].equals(aktNev)) {
                                j++;
                        }
                        if (j <= tanuloListaDb) {
                                hianyzasDb[j - 1] += aktHianyzas;
                        } else {
                                tanuloListaDb++;
                                tanuloLista[tanuloListaDb - 1] = aktNev;
                                hianyzasDb[tanuloListaDb - 1] = aktHianyzas;
                        }
                }
                // A maximális hiányzás meghatározása
                int maxHianyzas = hianyzasDb[0];
                for (i = 2; i <= tanuloListaDb; i++) {
                        if (hianyzasDb[i - 1] > maxHianyzas) {
                                maxHianyzas = hianyzasDb[i - 1];
                        }
                }

                // A legtöbbet hiányzó tanulók kiválogatása
                for (i = 1; i <= tanuloListaDb; i++) {
                        if (hianyzasDb[i - 1] == maxHianyzas) {
                                System.out.print(tanuloLista[i - 1] + " ");
                        }
                }
                System.out.println("\n");

        }

        public static String hetnapja(int honap, int nap) {
                String[] napnev = { "vasarnap", "hetfo", "kedd", "szerda", "csutortok", "pentek", "szombat" };
                int[] napszam = { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 335 };
                int napsorszam = (napszam[honap - 1] + nap) % 7;
                return napnev[napsorszam];
        }

}
```

```
Az 1. feladat megoldása
A naplo.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
A naplóban 139 bejegyzés van.

A 3. feladat megoldása
Az igazolt hiányzások száma 788,  az igazolatlanoké 18 óra.

A 4. feladat megoldása
A hetnapja függvény elkészítése

Az 5. feladat megoldása
Kérem egy hónap sorszámát (1 - 12): 2
Kérem a hónap napjának sorszámát (1 - 28/30/31): 3
Azon a napon szombat volt.

A 6. feladat megoldása
Kérem a hét egy napjának nevét (ékezetek nélkül): szerda
Kérem a tanítási óra sorszámát (1 - 7): 3
Az adott tanítási órára összesen 49 hiányzás jutott.

A 7. feladat megoldása
A legtöbbet hiányzó tanulók: Kivi Adrienn Jujuba Ibolya
```

2018 május: Társalgó

Egy színház társalgójában még a délelőtti próbák alatt is nagy a forgalom. A színészek hosszabb-rövidebb beszélgetésekre térnek be ide, vagy éppen csak keresnek valakit.
A feladatban a társalgó ajtajánál 9 és 15 óra között felvett adatokat kell feldolgoznia.

Az *ajto.txt* fájlban időrendben rögzítették, hogy ki és mikor lépett be vagy ki a társalgó egyetlen ajtaján. A fájl soraiban négy, szóközzel elválasztott érték található. Az első két szám az áthaladás időpontja (óra, perc), a harmadik a személy azonosítója, az utolsó az áthaladás iránya (be/ki). A sorok száma legfeljebb 1000, a személyek azonosítója egy 1 és 100 közötti egész szám. Biztosan tudjuk, hogy a megfigyelés kezdetén (9 órakor) a társalgó üres volt, de a megfigyelés végén (15 órakor) még lehettek benn a társalgóban. A társalgóba be- és kilépéseket azok sorrendjében tartalmazza az állomány, még akkor is, ha a perc pontossággal rögzített adatok alapján egyezőség áll fenn.
Például:

Fájl adatai	*Bentlévők száma*
9 1 2 be	1
9 1 9 be	2
9 3 15 be	3
9 5 9 ki	2
9 8 15 ki	1
9 8 20 be	2
9 8 26 be	3
9 13 4 be	4
9 13 26 ki	3
…	…

A fenti példában a szürke mintázatú részen a bemeneti fájl első néhány sora látható.
A második sora azt mutatja, hogy a 9-es azonosítójú személy 9 óra 1 perckor lépett be a társalgóba. A negyedik sorban olvasható, hogy 9 óra 5 perckor már ki is ment, tehát ekkor összesen 4 percet töltött bent. A szürke rész sorai mellett olvasható számok azt mutatják, hogy a be- vagy kilépést követően hányan vannak bent a társalgóban. Ez a szám egy percen belül akár többször is változhat.

Készítsen programot, amely az *ajto.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *tarsalgo* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `4. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el az *ajto.txt* fájl tartalmát!
2. Írja a képernyőre annak a személynek az azonosítóját, aki a vizsgált időszakon belül először lépett be az ajtón, és azét, aki utoljára távozott a megfigyelési időszakban!
3. Határozza meg a fájlban szereplő személyek közül, ki hányszor haladt át a társalgó ajtaján! A meghatározott értékeket azonosító szerint növekvő sorrendben írja az *athaladas.txt* fájlba! Soronként egy személy azonosítója, és tőle egy szóközzel elválasztva az áthaladások száma szerepeljen!

4. Írja a képernyőre azon személyek azonosítóját, akik a vizsgált időszak végén a társalgóban tartózkodtak!
5. Hányan voltak legtöbben egyszerre a társalgóban? Írjon a képernyőre egy olyan időpontot (óra:perc), amikor a legtöbben voltak bent!
6. Kérje be a felhasználótól egy személy azonosítóját!
   A további feladatok megoldásánál ezt használja fel!
   Feltételezheti, hogy a megadott azonosítóhoz tartozik adat a forrásfájlban.
7. Írja a képernyőre, hogy a beolvasott azonosítóhoz tartozó személy mettől meddig tartózkodott a társalgóban!
   A kiírást az alábbi, 22-es személyhez tartozó példának megfelelően alakítsa ki!

```
11:22-11:27
13:45-13:47
13:53-13:58
14:17-14:20
14:57-
```

8. Határozza meg, hogy a megfigyelt időszakban a beolvasott azonosítójú személy összesen hány percet töltött a társalgóban! Az előző feladatban példaként szereplő 22-es személy 5 alkalommal járt bent, a megfigyelés végén még bent volt. Róla azt tudjuk, hogy 18 percet töltött bent a megfigyelés végéig. A 39-es személy 6 alkalommal járt bent, a vizsgált időszak végén nem tartózkodott a helyiségben. Róla azt tudjuk, hogy 39 percet töltött ott. Írja ki, hogy a beolvasott azonosítójú személy mennyi időt volt a társalgóban, és a megfigyelési időszak végén bent volt-e még!

Minta a szöveges kimenetek kialakításához:

```
2. feladat
Az első belépő: 2
Az utolsó kilépő: 6

4. feladat
A végén a társalgóban voltak: 1 11 22 24 29 30 35 37

5. feladat
Például 10:44-kor voltak a legtöbben a társalgóban.

6. feladat
Adja meg a személy azonosítóját! 22

7. feladat
11:22-11:27
13:45-13:47
13:53-13:58
14:17-14:20
14:57-

8. feladat
A(z) 22. személy összesen 18 percet volt bent, a
megfigyelés végén a társalgóban volt.
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Társalgó
 *
 * @author Klemand
 */

public class EmeltInfo2018maj {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az ajto.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("ajto.txt"));

		int[] ora = new int[1000];
		int[] perc = new int[1000];
		int[] azon = new int[1000];
		String[] irany = new String[1000];
		int[] bentDb = new int[1000];

		int db = 0;
		bentDb[0] = 1; // Tudom, hogy először belépés történt.
		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			daraboltSor = sor.split(" ");
			db++;
			ora[db - 1] = Integer.parseInt(daraboltSor[0]);
			perc[db - 1] = Integer.parseInt(daraboltSor[1]);
			azon[db - 1] = Integer.parseInt(daraboltSor[2]);
			irany[db - 1] = daraboltSor[3];
			if (db > 1) {
				if (irany[db - 1].equals("be")) {
					bentDb[db - 1] = bentDb[db - 2] + 1;
				} else {
					bentDb[db - 1] = bentDb[db - 2] - 1;
				}
			}
		}

		beolvas.close();
		System.out.println("A beolvasás megtörtént.");
		System.out.println("Az ajtón összesen " + db + " esetben haladt át valaki.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Az első belépő: " + azon[0]);

		System.out.print("Az utolsó belépő: ");
		int i = db;
		while (i >= 1 && !(irany[i - 1].equals("ki"))) {
			i--;
		}
		if (i >= 1) {
			System.out.println(azon[i - 1] + "\n");
		} else {
			System.out.println("Senki sem lépett ki az ajtón.\n");
		}
```

```
System.out.println("A 3. feladat megoldása");

int[] at = new int[100]; // Az áthaladásokat számláló tömb
for (i = 1; i <= 100; i++) {
        at[i - 1] = 0;
}

int aktAzon;
for (i = 1; i <= db; i++) {
        aktAzon = azon[i - 1];
        at[aktAzon - 1]++;
}

PrintWriter kiir = new PrintWriter(new FileWriter("athaladas.txt"));

for (i = 1; i <= 100; i++) {
        if (at[i - 1] > 0) {
                kiir.println(i + " " + at[i - 1]);
        }
}

kiir.close();
System.out.println("A fájlkiírás megtörtént. \n");

System.out.println("A 4. feladat megoldása");

System.out.print("A végén a társalgóban voltak: ");

for (i = 1; i <= 100; i++) {
        if (at[i - 1] % 2 == 1) { // Páratlanszor haladt át az ajtón
                System.out.print(i + " ");
        }
}
System.out.println("\n");

System.out.println("Az 5. feladat megoldása");

System.out.print("Például ");

int max = 1;
for (i = 2; i <= db; i++) {
        if (bentDb[i - 1] > bentDb[max - 1]) {
                max = i;
        }
}

System.out.print(ora[max - 1] + ":" + perc[max - 1] + "-kor voltak legtöbben a társalgóban: ");
System.out.println(bentDb[max - 1] + " személy\n");

System.out.println("A 6. feladat megoldása");

System.out.print("Adja meg a személy azonosítóját! ");

Scanner sc = new Scanner(System.in);
aktAzon = sc.nextInt();

sc.close();
System.out.println();
```

```
                System.out.println("A 7. feladat megoldása");

                for (i = 1; i <= db; i++) {
                        if (azon[i - 1] == aktAzon) {
                                if (irany[i - 1].equals("be")) {
                                        System.out.print(ora[i - 1] + ":" + perc[i - 1] + "-");
                                } else {
                                        System.out.println(ora[i - 1] + ":" + perc[i - 1]);
                                }
                        }
                }

                System.out.println();
                if (at[aktAzon - 1] % 2 == 1) {
                        System.out.println();
                }

                System.out.println("A 8. feladat megoldása");

                int osszIdo = 0;
                for (i = 1; i <= db; i++) {
                        if (azon[i - 1] == aktAzon) {
                                if (irany[i - 1].equals("be")) {
                                        osszIdo -= ido(ora[i - 1], perc[i - 1]);
                                } else {
                                        osszIdo += ido(ora[i - 1], perc[i - 1]);
                                }
                        }
                }

                if (at[aktAzon - 1] % 2 == 1) {
                        osszIdo += ido(15, 0);
                }

                System.out.print("A(z) " + aktAzon + ". személy összesen " + osszIdo + " percet volt bent, ");
                if (at[aktAzon - 1] % 2 == 1) {
                        System.out.println("a megfigyelés végén a társalgóban volt.\n");
                } else {
                        System.out.println("a megfigyelés végén nem volt a társalgóban.\n");
                }

        }

        public static int ido(int o, int p) {
                return 60 * o + p;
        }

}
```

```
Az 1. feladat megoldása
Az ajto.txt fájl beolvasása
A beolvasás megtörtént.
Az ajtón összesen 516 esetben haladt át valaki.

A 2. feladat megoldása
Az első belépő: 2
Az utolsó belépő: 6

A 3. feladat megoldása
A fájlkiírás megtörtént.

A 4. feladat megoldása
A végén a társalgóban voltak: 1 11 22 24 29 30 35 37

Az 5. feladat megoldása
Például 10:44-kor voltak legtöbben a társalgóban: 12 személy

A 6. feladat megoldása
Adja meg a személy azonosítóját! 39

A 7. feladat megoldása
9:57-10:0
10:50-11:0
11:24-11:28
11:45-12:3
12:34-12:36
12:47-12:49

A 8. feladat megoldása
A(z) 39. személy összesen 39 percet volt bent, a megfigyelés végén nem volt a társalgóban.
```

Az athaladas.txt szövegfájl:

```
1 21
2 12
3 14
4 14
5 10
6 22
7 18
9 12
10 18
11 13
12 8
14 10
15 14
16 16
18 10
20 10
21 16
22 9
23 22
24 15
25 32
26 10
27 14
28 10
29 13
30 11
31 16
32 8
35 21
36 26
37 17
38 18
39 12
40 12
```

2018 május idegennyelvű: Fogadóóra

Egy iskolában a tanárok fogadóóráira egy webes felületen foglalhatnak időpontot a szülők. Ebben a feladatban az egyik fogadónap adataival kell dolgoznia. A fogadónap 16:00-tól 18:00-ig tart, a lehetséges lefoglalható időpontok: 16:00, 16:10, 16:20 … 17:50. Egy-egy megbeszélés 10 percig tart. Időpontütközést a foglalást felügyelő program nem enged meg.

A *fogado.txt* fájl a tanárok foglaltsági adatait tartalmazza. Egy sorban a következő adatok találhatók szóközzel elválasztva: a tanár vezetékneve; utóneve; a lefoglalt időpont; a foglalás rögzítésének dátuma és időpontja. A tanár neve pontosan egy vezetéknévből és pontosan egy utónévből áll. Az óra, perc, hónap és nap adatok mindegyikét pontosan két számjeggyel tárolva találhatja meg a fájlban. A fájlban biztosan 500-nál kevesebb sor fordul elő, és az adatok sorrendje véletlenszerű.

Például:

```
…
Nagy Marcell 16:30 2017.10.29-20:32
Fodor Zsuzsanna 17:10 2017.10.28-23:12
Lakatos Levente 16:00 2017.10.30-08:24
…
```

A példa első sora szerint Nagy Marcell tanár úrnál a 16:30-as időpontot lefoglalták, mégpedig 2017. 10. 29-én 20:32-kor.

Készítsen programot, amely a `fogado.txt` állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `fogado` néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például `2. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el a `fogado.txt` fájl tartalmát!
2. Írja a képernyőre, hogy hány foglalás adatait tartalmazza a fájl!
3. Kérje be a felhasználótól egy tanár nevét, majd jelenítse meg a mintának megfelelően a képernyőn, hogy a megadott tanárnak hány időpontfoglalása van! Ha a megadott tanárhoz – ilyen például Farkas Attila – még nem történt foglalás, akkor „A megadott néven nincs időpontfoglalás.” üzenetet jelenítse meg!
4. Kérjen be a felhasználótól egy érvényes időpontot a forrásfájlban található formátumban (pl. 17:40)! A program írja a képernyőre a megadott időpontban foglalt tanárok névsorát! Egy sorban egy név szerepeljen! A névsor ábécé szerint rendezett legyen! A rendezett névsort írja ki fájlba is, és ott is soronként egy név szerepeljen! Az időpontnak megfelelő fájlnevet használjon, például 17:40 esetén a *1740.txt* fájlban tárolja el az adatokat! Ügyeljen arra, hogy a fájlnév a kettőspont karaktert ne tartalmazza! (Amennyiben ezen a néven nem tudja a fájlt létrehozni, használja az `adatok.txt` állománynevet!)
5. Határozza meg, majd írja ki a képernyőre a legkorábban lefoglalt időpont minden adatát! Az adatok megjelenítésénél pontosan kövesse a feladat végén szereplő mintát!

6. Írja ki a képernyőre „***Barna Eszter***” tanárnő szabad időpontjait! Tudjuk, hogy a tanárnőnek legalább egy foglalt és több szabad időpontja is van. A tanárnő a legutolsó szülő fogadása után távozhat az iskolából. Mikor távozhat legkorábban? Az időpontot azonosíthatóan írja ki a képernyőre!

Minta a szöveges kimenetek kialakításához:

```
2. feladat
Foglalások száma: 161

3. feladat
Adjon meg egy nevet: Nagy Ferenc
Nagy Ferenc néven 6 időpontfoglalás van.

4. feladat
Adjon meg egy érvényes időpontot (pl. 17:10): 17:40
Beke Bianka
Csorba Ede
Fodor Zsuzsanna
Hantos Hedvig
Keller Katalin
Magos Magdolna
Nagy Marcell
Olasz Ferenc
Papp Lili
Szalai Levente
Veres Gergely

5. feladat
Tanár neve: Csorba Ede
Foglalt időpont: 16:30
Foglalás ideje: 2017.10.28-18:48

6. feladat
16:00
16:10
17:00
17:40
17:50
Barna Eszter legkorábban távozhat: 17:40
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Fogadóóra
 *
 * @author Klemand
 */

public class EmeltInfo2018mid {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A fogado.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("fogado.txt"));

		String[] nev = new String[500];
		String[] foglaltIdo = new String[500];
		String[] foglalas = new String[500];

		int db = 0;
		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			daraboltSor = sor.split(" ");
			db++;
			nev[db - 1] = daraboltSor[0] + " " + daraboltSor[1];
			foglaltIdo[db - 1] = daraboltSor[2];
			foglalas[db - 1] = daraboltSor[3];
		}

		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("A foglalások száma: " + db + "\n");

		System.out.println("A 3. feladat megoldása");
		System.out.print("Adjon meg egy nevet: ");

		Scanner sc = new Scanner(System.in);
		String aktNev = sc.nextLine();

		int aktDb = 0;
		int i;
		for (i = 1; i <= db; i++) {
			if (nev[i - 1].equals(aktNev)) {
				aktDb++;
			}
		}

		if (aktDb > 0) {
			System.out.println(aktNev + " néven " + aktDb + " foglalás van.\n");
		} else {
			System.out.println("A megadott néven nincs időpontfoglalás.\n");
		}
```

```
System.out.println("A 4. feladat megoldása");

System.out.print("Adjon meg egy érvényes időpontot (pl. 17:10): ");
String aktIdopont = sc.nextLine();

sc.close();

System.out.println("Foglalt tanárok:");

String[] aktFoglalt = new String[500];
aktDb = 0;
for (i = 1; i <= db; i++) {
        if (foglaltIdo[i - 1].equals(aktIdopont)) {
                aktDb++;
                aktFoglalt[aktDb - 1] = nev[i - 1];
        }
}

// Név szerinti buborékos rendezés, ékezetes betűket is megengedve!

int j;
String asztal;

for (i = aktDb - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (abcElobbNev(aktFoglalt[j], aktFoglalt[j - 1])) {

                        asztal = aktFoglalt[j - 1];
                        aktFoglalt[j - 1] = aktFoglalt[j];
                        aktFoglalt[j] = asztal;

                }
        }
}

String fileNev = aktIdopont.substring(0, 2) + aktIdopont.substring(3, 5) + ".txt";
PrintWriter kiir = new PrintWriter(new FileWriter(fileNev));

for (i = 1; i <= aktDb; i++) {
        System.out.println(aktFoglalt[i - 1]);
        kiir.println(aktFoglalt[i - 1]);
}

kiir.close();
System.out.println("A fájlkiírás megtörtént. \n");

System.out.println("Az 5. feladat megoldása");

System.out.println("A legkorábban lefoglalt időpont adatai: ");

// Most biztosan nincs ékezetes betű!

int min = 1;
for (i = 2; i <= db; i++) {
        if (foglalas[i - 1].compareTo(foglalas[min - 1]) < 0) {
                min = i;
        }
}

System.out.println("Tanár neve: " + nev[min - 1]);
System.out.println("Foglalt időpont: " + foglaltIdo[min - 1]);
System.out.println("Foglalas ideje: " + foglalas[min - 1] + "\n");
```

```
		System.out.println("A 6. feladat megoldása");
		System.out.println("Barna Eszter szabad időpontjai:");

		String[] idopontok = { "16:00", "16:10", "16:20", "16:30", "16:40", "16:50", "17:00", "17:10", "17:20", "17:30",
				"17:40", "17:50", "18:00" };
		int n = idopontok.length;
		boolean[] szabad = new boolean[n];
		for (i = 1; i <= n; i++) {
			szabad[i - 1] = true;
		}

		int aktIndex;
		for (i = 1; i <= db; i++) {
			if (nev[i - 1].equals("Barna Eszter")) {
				aktIndex = 1;
				while (!idopontok[aktIndex - 1].equals(foglaltIdo[i - 1])) {
					// Kiválasztás!
					aktIndex++;
				}
				szabad[aktIndex - 1] = false;
			}
		}

		for (i = 1; i <= n - 1; i++) {
			// 18:00 már nem foglalható!
			if (szabad[i - 1]) {
				System.out.println(idopontok[i - 1]);
			}
		}

		System.out.print("Barna Eszter legkorábban távozhat: ");
		i = n - 1;
		while (szabad[i - 1]) {
			// Tudjuk, hogy van foglalt időpontja!
			i--;
		}

		System.out.println(idopontok[i] + "\n");

	}

	public static boolean abcElobbNev(String nev1, String nev2) {
		String abc = " aábcdeéfghiíjklmnoóöőpqrstuúüűvwxyz";
		nev1 = nev1.toLowerCase();
		nev2 = nev2.toLowerCase();

		int k = 1;
		while (k <= nev1.length() && k <= nev2.length() && nev1.substring(k - 1, k).equals(nev2.substring(k - 1, k))) {
			k++;
		}

		if (k <= nev1.length() && k <= nev2.length()) {
			return abc.indexOf(nev1.substring(k - 1, k)) < abc.indexOf(nev2.substring(k - 1, k));
		} else {
			return nev1.length() < nev2.length();
		}
	}

}
```

```
Az 1. feladat megoldása
A fogado.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
A foglalások száma: 161

A 3. feladat megoldása
Adjon meg egy nevet: Nagy Ferenc
Nagy Ferenc néven 6 foglalás van.

A 4. feladat megoldása
Adjon meg egy érvényes időpontot (pl. 17:10): 17:40
Foglalt tanárok:
Beke Bianka
Csorba Ede
Fodor Zsuzsanna
Hantos Hedvig
Keller Katalin
Magos Magdolna
Nagy Marcell
Olasz Ferenc
Papp Lili
Szalai Levente
Veres Gergely
A fájlkiírás megtörtént.

Az 5. feladat megoldása
A legkorábban lefoglalt időpont adatai:
Tanár neve: Csorba Ede
Foglalt időpont: 16:30
Foglalas ideje: 2017.10.28-18:48

A 6. feladat megoldása
Barna Eszter szabad időpontjai:
16:00
16:10
17:00
17:40
17:50
Barna Eszter legkorábban távozhat: 17:40
```

A 1740.txt szövegfile

```
Beke Bianka
Csorba Ede
Fodor Zsuzsanna
Hantos Hedvig
Keller Katalin
Magos Magdolna
Nagy Marcell
Olasz Ferenc
Papp Lili
Szalai Levente
Veres Gergely
```

2018 október: Kerítés (Utca)

Egy üdülőfalu újonnan nyitott utcájában a telkeket a saroktól kiindulva egymás után folyamatosan, kihagyások nélkül adják el. A vásárló kiválaszthatja az oldalt, amelyen vásárolni akar (ott csak a soron következő telket vásárolhatja meg), valamint megadhatja a telek utcafronti szélességét. Sok telket vettek meg az utcában, a legtöbben már kerítést is építettek, azok majd' mindegyikét be is festették.

A *kerites.txt* fájl az utca telkeinek jelenlegi állapotát írja le. A telkek a vásárlás sorrendjében szerepelnek. Minden sorban három adat található. Az első szám megadja, hogy a telek a páros (0) vagy a páratlan (1) oldalán van az utcának; a második a telek szélességét adja meg méterben (egész szám, értéke 8 és 20 között lehet); a harmadik pedig az utcafronti kerítés színét leíró karakter. A szín az angol ábécé nagybetűje. Ha a kerítést már elkészítették, de nem festették be, akkor a „#" karakter, ha még nem készült el, akkor a „:" (kettőspont) karakter szerepel. Az utca hossza legfeljebb 1000 méter. Mindkét oldalon elkelt legalább 3-3 telek.
Például:

```
0 10 P
1 8 K
1 10 :
1 9 S
0 10 P
…
```

Az első telket a páros oldalon vették (házszáma: 2), 10 méter széles és már a kerítés is elkészült, amelyet P színnel festettek be. A második vásárló az első, aki a páratlan oldalon vett telket (házszáma: 1), 8 méter széles, K színű kerítése van. A harmadik vásárló is a páratlan oldalt választotta, ezért házszáma 3, 10 méteres a telke, de a kerítés még nem készült el.

Készítsen programot, amely a *kerites.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *utca* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például 5. feladat)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el a *kerites.txt* fájl tartalmát!
2. Írja a képernyőre, hogy hány telket adtak el az utcában!
3. Jelenítse meg a képernyőn, hogy az utolsó eladott telek
   a. melyik (páros / páratlan) oldalon talált gazdára!
   b. milyen házszámot kapott!
4. Írjon a képernyőre egy házszámot a páratlan oldalról, amely melletti telken ugyanolyan színű a kerítés! (A hiányzó és a festetlen kerítésnek nincs színe.) Feltételezheti, hogy van ilyen telek, a több ilyen közül elég az egyik ház számát megjeleníteni.
5. Kérje be a felhasználótól egy eladott telek házszámát, majd azt felhasználva oldja meg a következő feladatokat!
   a. Írja ki a házszámhoz tartozó kerítés színét, ha már elkészült és befestették, egyébként az állapotát a „**#**" vagy „**:**" karakter jelöli!
   b. A házszámhoz tartozó kerítést szeretné tulajdonosa be- vagy átfesteni. Olyan színt akar választani, amely különbözik a mellette lévő szomszéd(ok)tól és a jelenlegi színtől is. Adjon meg egy lehetséges színt! A színt a teljes palettából (A–Z) szabadon választhatja meg.

6. Jelenítse meg az *utcakep.txt* fájlban a páratlan oldal utcaképét az alábbi mintának megfelelően!

```
KKKKKKKK::::::::::SSSSSSSSSBBBBBBBBFFFFFFFFFKKKKKKKKKIIIIIIII
1        3         5         7        9         11         13
```

Az első sorban a páratlan oldal jelenjen meg, a megfelelő méternyi szakasz kerítésszínét (vagy állapotát) jelző karakterrel! A második sorban a telek első karaktere alatt kezdődően a házszám álljon!

Minta a szöveges kimenetek kialakításához:

```
2. feladat
Az eladott telkek száma: 98
3. feladat
A páros oldalon adták el az utolsó telket.
Az utolsó telek házszáma: 78
4. feladat
A szomszédossal egyezik a kerítés színe: 73
5. feladat
Adjon meg egy házszámot! 83
A kerítés színe / állapota: A
Egy lehetséges festési szín: D
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Kerítés (Utca)
 *
 * @author Klemand
 */

public class EmeltInfo2018okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A kerites.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("kerites.txt"));

		int[] hsz = new int[252];
		int[] hossz = new int[252];
		String[] szin = new String[252];
		int db = 0;
		int hsz1 = -1;
		int hsz0 = 0;
		int paritas = 0;
		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			daraboltSor = sor.split(" ");
			paritas = Integer.parseInt(daraboltSor[0]);
			db++;
			if (paritas == 0) {
				hsz0 += 2;
				hsz[db - 1] = hsz0;
			} else {
				hsz1 += 2;
				hsz[db - 1] = hsz1;
			}
			hossz[db - 1] = Integer.parseInt(daraboltSor[1]);
			szin[db - 1] = daraboltSor[2];
		}

		beolvas.close();
		System.out.println("A beolvasás megtörtént.\n");

		System.out.println("A 2. feladat megoldása");
		System.out.println("Az eladott telkek száma: " + db);

		System.out.println("\nA 3. feladat megoldása");
		if (paritas == 0) {
			System.out.print("A páros ");
		} else {
			System.out.print("A páratlan ");
		}
		System.out.println("olalon adták el az utolsó telket.");
		System.out.println("Az utolsó telek házszáma " + hsz[db - 1]);
```

```
System.out.println("\nA 4. feladat megoldása");
String szinek = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
String egyezo = "";
int i, j;
for (i = 1; i < db; i++) {
        if (hsz[i - 1] % 2 == 1 && szinek.contains(szin[i - 1])) {
                j = i + 1;
                while (j <= db && hsz[j - 1] % 2 != 1) {
                        j++;
                }
                if (j <= db && szin[j - 1].equals(szin[i - 1])) {
                        egyezo += hsz[i - 1] + " ";
                }
        }
}
System.out.println("A következő szomszédjával egyezik a kerítés színe: " + egyezo);

System.out.println("\nAz 5. feladat megoldása");

System.out.print("Adjon meg egy házszámot! ");
Scanner sc = new Scanner(System.in);
int akthsz = sc.nextInt();
String tiltottSzin = "";
i = 1;
while (hsz[i - 1] < akthsz) {
        i++;
}

System.out.print("A kerítés színe / állapota: ");
if (szin[i - 1].equals(":")) {
        System.out.println("nem készült el.");
} else if (szin[i - 1].equals("#")) {
        System.out.println("nincs befestve.");
} else {
        System.out.println(szin[i - 1]);
}

tiltottSzin += szin[i - 1];

if (i > 1) {
        j = i - 1;
        while (j >= 1 && hsz[j - 1] != akthsz - 2) {
                j--;
        }
        if (j >= 1) {
                tiltottSzin += szin[j - 1];
        }
}

if (i < db) {
        j = i + 1;
        while (j <= db && hsz[j - 1] != akthsz + 2) {
                j++;
        }
        if (j <= db) {
                tiltottSzin += szin[j - 1];
        }
}

j = 1;
while (tiltottSzin.contains(szinek.substring(j - 1, j))) {
        j++;
}
System.out.println("Egy lehetséges festési szín: " + szinek.substring(j - 1, j));
sc.close();
```

```
                System.out.println("\nA 6. feladat megoldása");

                String elsoSor = "";
                String masodikSor = "";
                int szamhossz;
                for (i = 1; i <= db; i++) {
                        if (hsz[i - 1] % 2 == 1) {
                                for (j = 1; j <= hossz[i - 1]; j++) {
                                        elsoSor += szin[i - 1];
                                }
                                szamhossz = ("" + hsz[i - 1]).length();
                                masodikSor += hsz[i - 1];
                                for (j = 1; j <= hossz[i - 1] - szamhossz; j++) {
                                        masodikSor += " ";
                                }
                        }
                }

                PrintWriter kiir = new PrintWriter(new FileWriter("utcakep.txt"));
                kiir.println(elsoSor);
                kiir.println(masodikSor);

                kiir.close();
                System.out.println("A fájlkiírás megtörtént.");

        }

}
```

```
Az 1. feladat megoldása
A kerites.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
Az eladott telkek száma: 98

A 3. feladat megoldása
A páros olalon adták el az utolsó telket.
Az utolsó telek házszáma 78

A 4. feladat megoldása
A következő szomszédjával egyezik a kerítés színe: 73 93

Az 5. feladat megoldása
Adjon meg egy házszámot! 83
A kerítés színe / állapota: A
Egy lehetséges festési szín: D

A 6. feladat megoldása
A fájlkiírás megtörtént.
```

Az utcakep.txt szövegfile

```
KKKKKKKK::::::::::SSSSSSSSSBBBBBBBBFFFFFFFFFKKKKKKKKK…ZZZZZZZZZZZZZZZZZZZZVVVVVVVVVVVVVVVVVVVV
1       3         5        7       9        11        …115                 117
```

2019 május: Céges autók

Egy cég 10 olyan autóval rendelkezik, amelyet a dolgozók igénybe vehetnek az üzleti ügyeik intézésére. Az autókat akár többnapos útra is elvihetik, illetve egy autót egy nap több dolgozó is elvihet. A rendszer az autók parkolóból való ki- és behajtását rögzíti. A parkoló a hónap minden napján 7-23 óra között van nyitva, csak ebben az időszakban lehet elvinni és visszahozni az autókat. Az autót mindig annak a dolgozónak kell visszahoznia, amelyik elvitte. Egyszerre csak egy autó lehet minden dolgozónál.

Az *autok.txt* fájl egy hónap (30 nap) adatait rögzíti. Egy sorban szóközökkel elválasztva 6 adat található az alábbi sorrendben.

```
nap, egész szám (1-30), a hónap adott napja
óra:perc, szöveg (óó:pp formátumban), a ki- vagy a behajtás időpontja
rendszám, 6 karakteres szöveg (CEG300-CEG309), az autó rendszáma
személy azonosítója, egész szám (500-600), az autót igénybe vevő dolgozó azonosítója
km számláló, egész szám, a km számláló állása
ki/be hajtás, egész szám (0 vagy 1), a parkolóból kihajtáskor 0, a behajtáskor 1
```

A sorok száma legfeljebb 500. Az adatok a napok szerint, azon belül óra és perc szerint rendezettek. Továbbá tudjuk, hogy a hónap első napján a cég mind a tíz autója a parkolóban volt.
Például:

```
…
5 07:30 CEG300 590 30580 0
5 14:16 CEG300 590 30656 1
5 17:00 CEG300 534 30656 0
5 19:03 CEG300 534 30784 1
…
15 09:53 CEG308 543 35048 0
17 11:16 CEG308 543 35746 1
```

A példában látható, hogy a CEG300 rendszámú autót az 5. napon kétszer is elvitték. Először 7:30-kor vitték el és 14:16-kor hozta vissza az 590-es dolgozó. A kivitelkor a kilométerszámláló állása 30 580 km volt, amikor visszahozta 30 656 km volt. Másodszor 17:00-kor vitte el az 534-es dolgozó az autót és 19:03-kor hozta vissza. A CEG308 rendszámú autót pedig a 15. napon vitte el az 543-as dolgozó és a 17. napon hozta vissza.
Készítsen programot, amely az *autok.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *cegesauto* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)
A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.
Az eredmény megjelenítését és a felhasználóval való kommunikációt a feladatot követő minta alapján valósítsa meg!

1. Olvassa be és tárolja el az *autok.txt* fájl tartalmát!
2. Adja meg, hogy melyik autót vitték el utoljára a parkolóból! Az eredményt a mintának megfelelően írja a képernyőre!
3. Kérjen be egy napot és írja ki a képernyőre a minta szerint, hogy mely autókat vitték ki és hozták vissza az adott napon!
4. Adja meg, hogy hány autó nem volt bent a hónap végén a parkolóban!
5. Készítsen statisztikát, és írja ki a képernyőre mind a 10 autó esetén az ebben a hónapban megtett távolságot kilométerben! A hónap végén még kint lévő autók esetén az utolsó rögzített kilométerállással számoljon! A kiírásban az autók sorrendje tetszőleges lehet.
6. Határozza meg, melyik személy volt az, aki az autó egy elvitele alatt a leghosszabb távolságot tette meg! A személy azonosítóját és a megtett kilométert a minta szerint írja a képernyőre! (Több legnagyobb érték esetén bármelyiket kiírhatja.)
7. Az autók esetén egy havi menetlevelet kell készíteni! Kérjen be a felhasználótól egy rendszámot! Készítsen egy *X_menetlevel.txt* állományt, amelybe elkészíti az adott rendszámú autó menetlevelét! (Az X helyére az autó rendszáma kerüljön!) A fájlba soronként tabulátorral elválasztva a személy azonosítóját, a kivitel időpontját (nap. óra:perc), a kilométerszámláló állását, a visszahozatal időpontját (nap. óra:perc), és a kilométerszámláló állását írja a minta szerint! (A tabulátor karakter ASCII-kódja: 9.)

Minta a szöveges kimenetek kialakításához:

```
2. feladat
30. nap rendszám: CEG300
3. feladat
Nap: 4
Forgalom a(z) 4. napon:
12:50 CEG303 561 ki
19:17 CEG308 552 be
4. feladat
A hónap végén 4 autót nem hoztak vissza.
5. feladat
CEG300 6751 km
CEG301 5441 km
CEG302 5101 km
CEG303 7465 km
CEG304 6564 km
CEG305 5232 km
CEG306 7165 km
CEG307 6489 km
CEG308 6745 km
CEG309 1252 km
6. feladat
Leghosszabb út: 1551 km, személy: 506
7. feladat
Rendszám: CEG304
Menetlevél kész.
```

A *CEG304_menetlevel.txt* fájl tartalma:

```
...
588 21. 16:58 13452 km 23. 20:28 14335 km
512 24. 16:58 14335 km 26. 22:21 15041 km
504 27. 13:47 15041 km
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class EmeltInfo2019maj {
	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az autok.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("autok.txt"));

		int[] nap = new int[500];
		String[] ido = new String[500];
		int[] azon = new int[500];
		int[] km = new int[500];
		String[] rsz = new String[500];
		int[] kibe = new int[500]; // 0 ki, 1 be

		int db = 0;
		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			daraboltSor = sor.split(" ");
			db++;
			nap[db - 1] = Integer.parseInt(daraboltSor[0]);
			ido[db - 1] = daraboltSor[1];
			rsz[db - 1] = daraboltSor[2];
			azon[db - 1] = Integer.parseInt(daraboltSor[3]);
			km[db - 1] = Integer.parseInt(daraboltSor[4]);
			kibe[db - 1] = Integer.parseInt(daraboltSor[5]);
		}

		beolvas.close();

		System.out.println("A 2. feladat megoldása");

		int i = db;
		while (kibe[i - 1] == 1 && i >= 1) {
			i--;
		}
		if (i >= 1) {
			System.out.println(nap[i - 1] + ". nap rendszám: " + rsz[i - 1]);

		} else {
			System.out.println("Egy autót sem vittek el.");
		}

		System.out.println("A 3. feladat megoldása");

		Scanner sc1 = new Scanner(System.in);
		System.out.print("Nap: ");
		int aktNap = sc1.nextInt();
		System.out.println("Forgalom a(z) " + aktNap + ". napon: ");
```

```
for (i = 1; i <= db; i++) {
        if (nap[i - 1] == aktNap) {
                System.out.print(ido[i - 1] + " " + rsz[i - 1] + " " + azon[i - 1]);
                if (kibe[i - 1] == 1) {
                        System.out.println(" be");
                } else {
                        System.out.println(" ki");
                }
        }
}

System.out.println("A 4. feladat megoldása");

boolean[] bentVan = new boolean[10];
for (i = 0; i <= 9; i++) {
        bentVan[i] = true; // Kezdetben mind bent van.
}

int aktAuto;
for (i = 1; i <= db; i++) {
        aktAuto = Integer.parseInt(rsz[i - 1].substring(5));
        bentVan[aktAuto] = (kibe[i - 1] == 1);
}

int dbKint = 0;
for (i = 0; i <= 9; i++) {
        if (!bentVan[i]) {
                dbKint++;
        }
}
System.out.println("A hónap végén " + dbKint + " autót nem hoztak vissza.");

System.out.println("Az 5. feladat megoldása");

int elsoKm[] = new int[10];
int osszKm[] = new int[10];

for (i = 0; i <= 9; i++) {
        elsoKm[i] = 0;//Vizsgálathoz kell
        osszKm[i] = 0;//Ha egyszer sem vitték ki, akkor csak itt kap értéket
}
for (i = 1; i <= db; i++) {
        aktAuto = Integer.parseInt(rsz[i - 1].substring(5));
        if (elsoKm[aktAuto] == 0) {
                elsoKm[aktAuto] = km[i - 1];
        } else {
                if (kibe[i - 1] == 1) {
                        osszKm[aktAuto] = km[i - 1] - elsoKm[aktAuto];
                }
        }
}

for (i = 0; i <= 9; i++) {
        System.out.println("CEG30" + i + " " + osszKm[i] + " km");
}

System.out.println("A 6. feladat megoldása");

int[] aktKm = new int[10];
int[] maxKm = new int[10];
int[] maxAzon = new int[10];
```

```
                for (i = 0; i <= 9; i++) {
                        maxKm[i] = 0; // Kezdőértékek
                        maxAzon[i] = -1; // irreális kezdőérték
                }

                for (i = 1; i <= db; i++) {
                        aktAuto = Integer.parseInt(rsz[i - 1].substring(5));
                        if (kibe[i - 1] == 0) {
                                aktKm[aktAuto] = km[i - 1] * (-1);
                        } else {
                                aktKm[aktAuto] += km[i - 1];
                                if (aktKm[aktAuto] > maxKm[aktAuto]) {
                                        maxKm[aktAuto] = aktKm[aktAuto];
                                        maxAzon[aktAuto] = azon[i - 1];
                                }
                        }
                }

                int maxUt = maxKm[0];
                int maxSzemely = maxAzon[0];
                for (i = 1; i <= 9; i++) {
                        if (maxKm[i] > maxUt) {
                                maxUt = maxKm[i];
                                maxSzemely = maxAzon[i];
                        }
                }

                System.out.println("Leghosszabb út: " + maxUt + " km, személy: " + maxSzemely);

                System.out.println("A 7. feladat megoldása");

                Scanner sc2 = new Scanner(System.in);
                System.out.print("Rendszám: ");
                String aktRsz = sc2.nextLine();

                sc1.close();
                sc2.close();

                String cim = "" + aktRsz + "_menetlevel.txt";
                PrintWriter kiir = new PrintWriter(new FileWriter(cim));

                for (i = 1; i <= db; i++) {
                        if (rsz[i - 1].equals(aktRsz)) {
                                if (kibe[i - 1] == 0) {
                                        kiir.print(azon[i - 1] + "\t" + nap[i - 1] + ". \t" + ido[i - 1] + "\t" +
                                                km[i - 1] + " km \t");
                                } else {
                                        kiir.println(nap[i - 1] + ". \t" + ido[i - 1] + "\t" + km[i - 1] + " km");
                                }
                        }
                }

                System.out.println("Menetlevél kész.");
                kiir.close();
        }
}
```

```
Az 1. feladat megoldása
Az autok.txt fájl beolvasása
A 2. feladat megoldása
30. nap rendszám: CEG300
A 3. feladat megoldása
Nap: 4
Forgalom a(z) 4. napon:
12:50 CEG303 561 ki
19:17 CEG308 552 be
A 4. feladat megoldása
A hónap végén 4 autót nem hoztak vissza.
Az 5. feladat megoldása
CEG300 6751 km
CEG301 5441 km
CEG302 5101 km
CEG303 7465 km
CEG304 6564 km
CEG305 5232 km
CEG306 7165 km
CEG307 6489 km
CEG308 6745 km
CEG309 1252 km
A 6. feladat megoldása
Leghosszabb út: 1551 km, személy: 506
A 7. feladat megoldása
Rendszám: CEG304
Menetlevél kész.
```

Az CEG304_menetlevel.txt szövegfile

```
583     1.      09:04   8477 km         2.      22:35   9235 km
516     3.      14:08   9235 km         3.      16:04   9414 km
551     3.      17:53   9414 km         3.      22:48   9569 km
546     5.      08:20   9569 km         5.      14:22   9652 km
517     5.      15:04   9652 km         9.      19:34   11086 km
551     10.     12:49   11086 km        11.     21:20   11999 km
574     13.     10:12   11999 km        13.     20:17   12153 km
549     15.     14:46   12153 km        16.     21:29   12854 km
591     18.     08:43   12854 km        18.     18:54   13007 km
534     19.     07:53   13007 km        19.     22:43   13279 km
560     20.     08:35   13279 km        20.     11:27   13452 km
588     21.     16:58   13452 km        23.     20:28   14335 km
512     24.     16:58   14335 km        26.     22:21   15041 km
504     27.     13:47   15041 km
```

2019 május idegen nyelvű: Tantárgyfelosztás

A tantárgyfelosztás a tanév tervezésének alapvető dokumentuma. A tantárgyfelosztás azt tartalmazza, hogy a tanárok a tantárgyaikat mely osztályokban, hány órában tanítják. Ebben a feladatban egy négy évfolyamos gimnázium tantárgyfelosztásának adatait kell elemeznie.

A tantárgyfelosztást ezúttal egy adatbázis-kezelő programmal előállított, egyszerű szerkezetű szöveges állományban kapja az alábbi minta szerint (Minden bejegyzést négy sor tárol.):

```
Albatrosz Aladin
biologia
9.a
2
Albatrosz Aladin
osztalyfonoki
9.a
1
```

…

```
Csincsilla Csilla
matematika
9.x
2
```

…

Az első bejegyzés megadja, hogy Albatrosz Aladin tanár úr biológiát (biologia) fog tanítani a 9.a osztályban heti 2 órában. Ha az osztály betűjele x, akkor évfolyam szintű csoportról van szó. Példánkban Csincsilla Csilla tanárnő a 9. évfolyam részére heti 2 órás matematika órát tart. Az osztályfőnököket arról ismerhetjük fel, hogy ők tartják az osztályfőnöki (osztalyfonoki) órát.

A megoldás során felhasználhatja, hogy a fájl maximum 1000 bejegyzést (azaz legfeljebb 4000 sort) tartalmaz. Az iskolában legfeljebb 100 tanár és legfeljebb 50 osztály van, továbbá minden osztálynak pontosan egy osztályfőnöke van.

Készítsen programot, amely a `beosztas.txt` állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `tanfel` néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, és feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok esetén – a mintához tartalmában hasonlóan – írja ki a képernyőre a feladat sorszámát (például: `3. feladat:`), és utaljon a kiírt tartalomra is!

Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Mindkét esetben az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el a *beosztas.txt* állományban talált adatokat, és annak felhasználásával oldja meg a következő feladatokat!
2. Hány bejegyzés található az állományban? Az eredményt írassa ki a képernyőre!
3. A fenntartó számára fontos információ, hogy az iskolában hetente összesen hány tanítási óra van. Határozza meg ezt az adatot és írassa ki a képernyőre!
4. Kérje be a felhasználótól egy tanár nevét, és írassa ki a képernyőre, hogy hetente hány órában tanít!
5. Készítse el az *of.txt* fájlt, amely az osztályfőnökök nevét tartalmazza osztályonként az alábbi formában (az osztályok megjelenítésének sorrendje a mintától eltérhet):

```
9.a - Albatrosz Aladin
9.b - Hangya Hanna
9.c - Zerge Zenina
…
```

6. Egyes osztályokban bizonyos tantárgyakat a tanulók csoportbontásban tanulnak: ekkor az adott tantárgyra és osztályra két bejegyzést is tartalmaz a tantárgyfelosztás. Kérje be egy osztály azonosítóját, valamint egy tantárgy nevét, és írassa ki a képernyőre, hogy az adott osztály a megadott tantárgyat csoportbontásban vagy osztályszinten tanulja-e!
   (Feltételezheti, hogy a megadott osztály tanulja a megadott tantárgyat.)
7. A fenntartó számára az is fontos információ, hogy hány tanár dolgozik az iskolában. Írassa ki ezt az adatot a képernyőre!

Példa a szöveges kimenetek kialakításához:

```
2. feladat
A fájlban 329 bejegyzés van.
3. feladat
Az iskolában a heti összóraszám: 1016
4. feladat
Egy tanár neve= Albatrosz Aladin
A tanár heti óraszáma: 24
6. feladat
Osztály= 10.b
Tantárgy= kemia
Csoportbontásban tanulják.
7. feladat
Az iskolában 49 tanár tanít.
```

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class EmeltInfo2019mid {
	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A beosztas.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("beosztas.txt"));

		String[] tanar = new String[1000];
		String[] tantargy = new String[1000];
		String[] osztaly = new String[1000];
		int[] ora = new int[1000];

		int db = 0;
		String sor;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			tanar[db - 1] = sor;
			tantargy[db - 1] = beolvas.readLine();
			osztaly[db - 1] = beolvas.readLine();
			ora[db - 1] = Integer.parseInt(beolvas.readLine());
		}

		beolvas.close();

		System.out.println("A 2. feladat megoldása");

		System.out.println("A fájlban " + db + " bejegyzés van. ");

		System.out.println("A 3. feladat megoldása");

		int osszOra = 0;
		int i;
		for (i = 1; i <= db; i++) {
			osszOra += ora[i - 1];
		}
		System.out.println("Az iskolában a heti óraszám:  " + osszOra);

		System.out.println("A 4. feladat megoldása");

		Scanner sc = new Scanner(System.in);
		System.out.print("Egy tanár neve= ");
		String aktTanar = sc.nextLine();
		int tanarOra = 0;

		for (i = 1; i <= db; i++) {
			if (tanar[i - 1].equals(aktTanar)) {
				tanarOra += ora[i - 1];
			}
		}

		System.out.println("A tanár heti óraszáma:  " + tanarOra);
```

```
System.out.println("Az 5. feladat megoldása");

PrintWriter kiir = new PrintWriter(new FileWriter("of.txt"));

String[] osztalyLista = new String[50];
int osztalyListaDb = 0;
String[] of = new String[50];

for (i = 1; i <= db; i++) {

        if (tantargy[i - 1].equals("osztalyfonoki")) {
                osztalyListaDb++;
                osztalyLista[osztalyListaDb - 1] = osztaly[i - 1];
                of[osztalyListaDb - 1] = tanar[i - 1];
        }
}

for (i = 1; i <= osztalyListaDb; i++) {
        kiir.println(osztalyLista[i - 1] + " - " + of[i - 1]);
}

System.out.println("A fájlkiírás megtörtént.");
kiir.close();

System.out.println("A 6. feladat megoldása");

String aktOsztaly;
String aktTantargy;

System.out.print("Osztály= ");
aktOsztaly = sc.nextLine();
System.out.print("Tantárgy= ");
aktTantargy = sc.nextLine();
sc.close();

int csoportDb = 0;

for (i = 1; i <= db; i++) {
        if (osztaly[i - 1].equals(aktOsztaly) && tantargy[i - 1].equals(aktTantargy)) {
                csoportDb++;
        }
}

if (csoportDb > 1) {
        System.out.println("Csoportbontásban tanulják.");
} else {
        System.out.println("Osztályszinten tanulják.");
}
```

```
                System.out.println("A 7. feladat megoldása");

                String[] tanarLista = new String[100];
                int tanarListaDb = 0;
                int j;

                for (i = 1; i <= db; i++) {
                        aktTanar = tanar[i - 1];
                        j = 1;
                        while (j <= tanarListaDb && !tanarLista[j - 1].equals(aktTanar)) {
                                j++;
                        }
                        if (j > tanarListaDb) {
                                tanarListaDb++;
                                tanarLista[tanarListaDb - 1] = aktTanar;
                        }
                }

                System.out.println("Az iskolában " + tanarListaDb + " tanár tanít.");

        }
}
```

```
Az 1. feladat megoldása
A beosztas.txt fájl beolvasása
A 2. feladat megoldása
A fájlban 329 bejegyzés van.
A 3. feladat megoldása
Az iskolában a heti óraszám:  1016
A 4. feladat megoldása
Egy tanár neve= Albatrosz Aladin
A tanár heti óraszáma:  24
Az 5. feladat megoldása
A fájlkiírás megtörtént.
A 6. feladat megoldása
Osztály= 10.b
Tantárgy= kemia
Csoportbontásban tanulják.
A 7. feladat megoldása
Az iskolában 49 tanár tanít.

Az of.txt szövegfile

9.a - Albatrosz Aladin
9.b - Hangya Hanna
9.c - Zerge Zenina
9.d - Medve Melani
10.a - Farkas Farkas
10.b - Bivaly Biti
10.c - Giliszta Gilbert
10.d - Borz Borka
11.a - Pekry Petra
11.b - Lemming Lea
11.c - Cet Celina
11.d - Panda Patrik
12.a - Kaffer Kada
12.b - Pulyka Pozsinka
12.c - Vidra Viktor
12.d - Puma Pongor
```

2019 október: eUtazás

Egyre több országban fordul elő, hogy a közlekedési eszközökön használatos bérleteket és jegyeket valamilyen elektronikus eszközön (például: chipes kártya) tárolják. Egy nagyváros ilyen rendszert szeretne bevezetni a helyi közlekedésben, amelyet néhány buszjáraton tesztelnek. Ezekre a buszokra csak az első ajtónál lehet felszállni, ahol egy ellenőrző eszközhöz kell érinteni a kártyát, amelynek chipje tartalmazza a jegy vagy bérlet információkat.

A busz ellenőrző eszköze statisztikai és fejlesztési célból rögzíti a felszállók kártyájának adatait. Az *utasadat.txt* szóközökkel tagolt állomány egy, a tesztelésben részt vevő busz végállomástól-végállomásig tartó útjának adatait tartalmazza.

Az *utasadat.txt* állomány legfeljebb 2000 sort tartalmaz és minden sorában 5 adat szerepel. Ezek:

- a megálló sorszáma (0-29; 0 az indulás helye és a 30 a végállomás, ahol már nem lehet felszállni.)
- a felszállás dátuma és időpontja (éééévhhnn-óópp formátumban, kötőjellel elválasztva a dátum és az idő)
- a kártya egyedi azonosítója (hétjegyű szám), egy utas a járaton legfeljebb egyszer utazik
- a jegy vagy bérlet típusa:

Azonosító	**Megnevezés**
FEB	Felnőtt bérlet
TAB	Tanulóbérlet (kedvezményes)
NYB	Nyugdíjas bérlet (kedvezményes)
NYP	65 év feletti bérlet (ingyenes)
RVS	Rokkant, vak, siket vagy kísérő bérlet (ingyenes)
GYK	Iskolakezdés előtti gyerekbérlet (ingyenes)
JGY	Jegy

- a bérlet érvényességi ideje, vagy a felhasználható jegyek száma. A bérlet esetén a dátum éééévhhnn formátumban szerepel, jegy esetén egy 0-10 közötti szám szerepel.

Például:

```
0 20190326-0700 6572582 RVS 20210101
0 20190326-0700 8808290 JGY 7
0 20190326-0700 1680423 TAB 20190420
12 20190326-0716 3134404 FEB 20190301
12 20190326-0716 9529716 JGY 0
```

A fenti példában szereplő adatoknál látható, hogy az induló állomáson (0. állomás) 2019. 03. 26-án 7:00-kor a 1680423 kártyaazonosítójú utas tanulóbérlettel szállt fel, amely 2019. 04. 20-ig érvényes. A 12. állomáson 2019. 03. 26-án 7:16-kor a 9529716 kártyaazonosítójú utas jeggyel szállt volna fel, de már elhasználta az összes jegyét (0).

Készítsen programot, amely az *utasadat.txt* állomány felhasználásával a következő kérdésekre válaszol! A program forráskódját *eutazas* néven mentse! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például `2. feladat`)! A részfeladatok eredményeit a mintán látható formában jelenítse meg! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el az *utasadat.txt* fájl tartalmát!

2. Adja meg, hogy hány utas szeretett volna felszállni a buszra!

3. A közlekedési társaság szeretné, ha a járművön csak az érvényes jeggyel vagy bérlettel rendelkezők utaznának. Ezért a jegyeket és bérleteket a buszvezető a felszálláskor ellenőrzi. (A bérlet még érvényes a lejárat napján.) Adja meg, hogy hány esetben kellett a buszvezetőnek elutasítania az utas felszállását, mert lejárt a bérlete vagy már nem volt jegye!

4. Adja meg, hogy melyik megállóban próbált meg felszállni a legtöbb utas! (Több azonos érték esetén a legkisebb sorszámút adja meg!)

5. A közlekedési társaságnak kimutatást kell készítenie, hogy hányszor utaztak valamilyen kedvezménnyel a járművön. Határozza meg, hogy hány kedvezményes és hány ingyenes utazó szállt fel a buszra! (Csak az érvényes bérlettel rendelkező szállhatott fel a buszra!)

6. Készítsen függvényt **napokszama** néven az alábbi algoritmus alapján. Az algoritmus a paraméterként megadott két dátumhoz (év, hónap, nap) megadja a közöttük eltelt napok számát! (A MOD a maradékos osztást, a DIV az egészrészes osztást jelöli.) Az algoritmust a *fuggveny.txt* fájlban is megtalálja. A függvényt a következő feladat megoldásához felhasználhatja.

```
Függvény napokszama(e1:egész, h1:egész, n1: egész, e2:egész,
                    h2: egész, n2: egész): egész
    h1 = (h1 + 9) MOD 12
    e1 = e1 - h1 DIV 10
    d1 = 365*e1 + e1 DIV 4 - e1 DIV 100 + e1 DIV 400 +
         (h1*306 + 5) DIV 10 + n1 - 1
    h2 = (h2 + 9) MOD 12
    e2 = e2 - h2 DIV 10
    d2 = 365*e2 + e2 DIV 4 - e2 DIV 100 + e2 DIV 400 +
         (h2*306 + 5) DIV 10 + n2 - 1
    napokszama:= d2-d1
Függvény vége
```

7. A közlekedési társaság azoknak az utasoknak, akiknek még érvényes, de 3 napon belül lejár a bérlete, figyelmeztetést szeretne küldeni e-mailben. (Például, ha a felszállás időpontja 2019. február 5., és a bérlet érvényessége 2019. február 8., akkor már kap az utas levelet, ha 2019. február 9. az érvényessége, akkor még nem kap levelet.) Válogassa ki és írja a *figyelmeztetes.txt* állományba ezen utasok kártyaazonosítóját és a bérlet érvényességi idejét (éééé-hh-nn formátumban) szóközzel elválasztva!

Minta a szöveges kimenetek kialakításához:

```
2. feladat
A buszra 699 utas akart felszállni.
3. feladat
A buszra 21 utas nem szállhatott fel.
4. feladat
A legtöbb utas (39 fő) a 8. megállóban próbált felszállni.
5. feladat
Ingyenesen utazók száma: 133 fő
A kedvezményesen utazók száma: 200 fő
```

Minta a *figyelmeztetes.txt* állomány kialakításához:

```
3023275 2019-03-29
2960983 2019-03-26
1581897 2019-03-27
2761792 2019-03-28
...
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * eUtazás
 *
 * @author Klemand
 */

public class EmeltInfo2019okt {

	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("Az utasadat.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("utasadat.txt"));

		int[] megallo = new int[2000];
		int[] datum = new int[2000];
		int[] ido = new int[2000];
		String[] azon = new String[2000];
		String[] tipus = new String[2000];
		int[] erv = new int[2000];

		int db = 0;
		String sor;
		String[] darSor;
		String[] darSor1;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			darSor = sor.split(" ");
			darSor1 = darSor[1].split("-");

			megallo[db - 1] = Integer.parseInt(darSor[0]);
			datum[db - 1] = Integer.parseInt(darSor1[0]);
			ido[db - 1] = Integer.parseInt(darSor1[1]);
			azon[db - 1] = darSor[2];
			tipus[db - 1] = darSor[3];
			erv[db - 1] = Integer.parseInt(darSor[4]);
		}

		beolvas.close();

		System.out.println("A beolvasás megtörtént.");

		System.out.println("\nA 2. feladat megoldása");

		System.out.println("A buszra " + db + " utas akart felszállni.");

		System.out.println("\nA 3. feladat megoldása");

		boolean[] ervenyes = new boolean[db];

		int i;
		for (i = 1; i <= db; i++) {
			ervenyes[i - 1] = true;
		}
```

```
for (i = 1; i <= db; i++) {
        if (tipus[i - 1].equals("JGY")) {
                if (erv[i - 1] == 0) {
                        ervenyes[i - 1] = false;
                }
        } else if (erv[i - 1] < datum[i - 1]) {
                ervenyes[i - 1] = false;
        }
}

int ervenytelenDb = 0;

for (i = 1; i <= db; i++) {
        if (!ervenyes[i - 1]) {
                ervenytelenDb++;
        }
}
System.out.println("A buszra " + ervenytelenDb + " utas nem szállhatott fel.");

System.out.println("\nA 4. feladat megoldása");

int[] utasDb = new int[30]; // Az egyes megállókban felszálló utasok száma

for (i = 0; i <= 29; i++) { // A megállók sorszáma 0-29
        utasDb[i] = 0; // Kezdőérték megadása
}

for (i = 1; i <= db; i++) {
        utasDb[megallo[i - 1]]++; // Érték szerinti indexelés
        // A megálló sorszámánál nem csúsztatunk!
}

// Maximumkiválasztás:

int max = 0;
// Abból indulunk ki, hogy a 0. megállóban volt a legtöbb felszálló.
for (i = 1; i <= 29; i++) {
        if (utasDb[i] > utasDb[max]) {
                max = i;
        }
}

System.out.println("A legtöbb utas (" + utasDb[max] + " fő) a " + max +
                   ". megállóban próbált felszállni.");

System.out.println("\nAz 5. feladat megoldása");

int ingyenesDb = 0;
int kedvezmenyesDb = 0;
String kedv = "TAB NYB";
String ingyenes = "NYP RVS GYK";
for (i = 1; i <= db; i++) {
        if (ervenyes[i - 1]) {
                if (kedv.contains(tipus[i - 1])) {
                        kedvezmenyesDb++;
                }
                if (ingyenes.contains(tipus[i - 1])) {
                        ingyenesDb++;
                }

        }
}

System.out.println("Ingyenesen utazók száma: " + ingyenesDb + " fő");
System.out.println("A kedvezményesen utazók száma: " + kedvezmenyesDb + " fő");
```

```
		System.out.println("\nA 6. feladat megoldása");
		System.out.println("A napokszama függvény elkészítése");
		// A függvény a main metódus után található

		System.out.println("\nA 7. feladat megoldása");

		PrintWriter kiir = new PrintWriter(new FileWriter("figyelmeztetes.txt"));

		int n1, h1, e1, n2, h2, e2;
		for (i = 1; i <= db; i++) {
			if (ervenyes[i-1] && !tipus[i - 1].equals("JGY")) {
				n1 = datum[i - 1] % 100;
				h1 = datum[i - 1] / 100;
				e1 = h1 / 100;
				h1 = h1 % 100;
				n2 = erv[i - 1] % 100;
				h2 = erv[i - 1] / 100;
				e2 = h2 / 100;
				h2 = h2 % 100;

				if (napokszama(e1, h1, n1, e2, h2, n2) <= 3) {
					kiir.println(azon[i - 1] + " " + e2 + "-" + h2 + "-" + n2);
				}
			}
		}

		kiir.close();
		System.out.println("A fájlkiírás megtörtént. \n");

	}

	public static int napokszama(int e1, int h1, int n1, int e2, int h2, int n2) {
		h1 = (h1 + 9) % 12;
		e1 = e1 - h1 / 10;
		int d1 = 365 * e1 + e1 / 4 - e1 / 100 + e1 / 400 + (h1 * 306 + 5) / 10 + n1 - 1;
		h2 = (h2 + 9) % 12;
		e2 = e2 - h2 / 10;
		int d2 = 365 * e2 + e2 / 4 - e2 / 100 + e2 / 400 + (h2 * 306 + 5) / 10 + n2 - 1;
		return d2 - d1;
	}

}
```

```
Az 1. feladat megoldása
Az utasadat.txt fájl beolvasása
A beolvasás megtörtént.

A 2. feladat megoldása
A buszra 699 utas akart felszállni.

A 3. feladat megoldása
A buszra 21 utas nem szállhatott fel.

A 4. feladat megoldása
A legtöbb utas (39 fő) a 8. megállóban próbált felszállni.

Az 5. feladat megoldása
Ingyenesen utazók száma: 133 fő
A kedvezményesen utazók száma: 200 fő

A 6. feladat megoldása
A napokszama függvény elkészítése

A 7. feladat megoldása
A fájlkiírás megtörtént.

A figyelmeztetes.txt szövegfile

2020534 2019-3-28
2416208 2019-3-29
3199016 2019-3-28
1446402 2019-3-27
4272667 2019-3-29
2734415 2019-3-27
2416648 2019-3-27
4043370 2019-3-26
4159578 2019-3-28
2896921 2019-3-27
3972945 2019-3-26
4764322 2019-3-27
1473856 2019-3-29
1618545 2019-3-26
4400156 2019-3-27
2223555 2019-3-27
2258830 2019-3-26
3023275 2019-3-29
2960983 2019-3-26
1581897 2019-3-27
2761792 2019-3-28
3702373 2019-3-26
3173542 2019-3-28
2496259 2019-3-29
4688169 2019-3-26
1062584 2019-3-28
3600757 2019-3-27
2190408 2019-3-27
3680746 2019-3-27
4658969 2019-3-27
4669550 2019-3-28
4108241 2019-3-26
4299308 2019-3-27
2373593 2019-3-29
1720477 2019-3-28
3380452 2019-3-27
3677994 2019-3-27
3552479 2019-3-27
```

2020 május: Meteorológiai jelentés

Az ország területén néhány városból rendszeres időközönként időjárás táviratokat küldenek. A távirat egy rövid szöveges üzenet, amely a főbb időjárási információkat tartalmazza. Rendelkezésünkre áll az ország területéről egy adott nap összes távirata.

A `tavirathu13.txt` szövegállomány egy adott hónap 13. napjának időjárás adataittartalmazza. Egy távirat adatai egy sorban találhatóak egymástól szóközzel elválasztva. Egy sorban 4 adat szerepel a következőképpen.

település	szöveg (2 karakter)	A település kétbetűs kódja
idő	szöveg (óópp formátumban)	A mérés időpontja
szélirány és -erősség	szöveg (5 karakter) szélirány 3 karakter, -erősség 2 karakter	A szél iránya fokban vagy szöveggel és sebessége csomóban megadva
hőmérséklet	egész szám (2 karakter)	Mért hőmérséklet (nem negatív)

A sorok száma legfeljebb 500. Az adatok idő szerint rendezettek.

Például:

```
BP 0300 32007 21
PA 0315 35010 19
PR 0315 32009 19
SM 0315 01015 20
DC 0315 VRB01 21
SN 0315 00000 21
```

A példában látható, hogy 03:15-kor PR településen 320 fokos irányból 9 csomós szél fújt.A hőmérséklet 19 °C volt. Ugyanekkor DC településen változó (VRB) szélirány volt 1 csomós szélsebességgel, a hőmérséklet 21 °C volt.

Készítsen programot, amely a `tavirathu13.txt` állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `metjelentes` néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `3. feladat`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

Az eredmény megjelenítését és a felhasználóval való kommunikációt a feladatot követő minta alapján valósítsa meg!

1. Olvassa be és tárolja el a *`tavirathu13.txt`* állomány adatait!
2. Kérje be a felhasználótól egy város kódját! Adja meg, hogy az adott városból mikor érkezett az utolsó mérési adat! A kiírásban az időpontot óó:pp formátumban jelenítse meg!
3. Határozza meg, hogy a nap során mikor mérték a legalacsonyabb és a legmagasabb hőmérsékletet! Jelenítse meg a méréshez kapcsolódó település nevét, az időpontot és a hőmérsékletet! Amennyiben több legnagyobb vagy legkisebb érték van, akkor elég az egyiket kiírnia.
4. Határozza meg, azokat a településeket és időpontokat, ahol és amikor a mérések idején szélcsend volt! (A szélcsendet a táviratban 00000 kóddal jelölik.) Ha nem volt ilyen, akkor a „`Nem volt szélcsend a mérések idején.`” szöveget írja ki! A kiírásnál a település kódját és az időpontot jelenítse meg.

5. Határozza meg a települések napi középhőmérsékleti adatát és a hőmérséklet-ingadozását! A kiírásnál a település kódja szerepeljen a sor elején a minta szerint! A kiírásnál csak a megoldott feladatrészre vonatkozó szöveget és értékeket írja ki!
    a. A középhőmérséklet azon hőmérsékleti adatok átlaga, amikor a méréshez tartozó óra értéke 1., 7., 13., 19. Ha egy településen a felsorolt órák valamelyikén nem volt mérés, akkor a kiírásnál az „NA” szót jelenítse meg! Az adott órákhoz tartozó összes adat átlagaként határozza meg a középhőmérsékletet, azaz minden értéket azonos súllyal vegyen figyelembe! A középhőmérsékletet egészre kerekítve jelenítse meg!
    b. A hőmérséklet-ingadozás számításhoz az adott településen a napi legmagasabb és legalacsonyabb hőmérséklet különbségét kell kiszámítania! (Feltételezheti, hogy minden település esetén volt legalább két mérési adat.)
6. Hozzon létre településenként egy szöveges állományt, amely első sorában a település kódját tartalmazza! A további sorokban a mérési időpontok és a hozzá tartozó szélerősségek jelenjenek meg! A szélerősséget a minta szerint a számértéknek megfelelő számú kettőskereszttel (#) adja meg! A fájlban az időpontokat és a szélerősséget megjelenítő kettőskereszteket szóközzel válassza el egymástól! A fájl neve *X.txt* legyen, ahol az X helyére a település kódja kerüljön!

Minta a szöveges kimenetek kialakításához:

```
2. feladat
Adja meg egy település kódját! Település: SM
Az utolsó mérési adat a megadott településről 23:45-kor érkezett.
3. feladat
A legalacsonyabb hőmérséklet: SM 23:45 16 fok.
A legmagasabb hőmérséklet: DC 13:15 35 fok.
4. feladat
BP 01:00
DC 02:15
SN 03:15
BC 04:45
DC 04:45
SN 05:15
SN 05:45
KE 08:45
BC 11:45
5. feladat
BP Középhőmérséklet: 23; Hőmérséklet-ingadozás: 8
DC Középhőmérséklet: 29; Hőmérséklet-ingadozás: 15
SM Középhőmérséklet: 22; Hőmérséklet-ingadozás: 8
PA Középhőmérséklet: 21; Hőmérséklet-ingadozás: 7
SN Középhőmérséklet: 26; Hőmérséklet-ingadozás: 13
PR Középhőmérséklet: 21; Hőmérséklet-ingadozás: 8
BC NA; Hőmérséklet-ingadozás: 14
PP NA; Hőmérséklet-ingadozás: 6
KE NA; Hőmérséklet-ingadozás: 13
6. feladat
A fájlok elkészültek.
```

A `BC.txt` fájl tartalma:

```
BC
00:45 ###
01:45 ####
02:45 ######
03:45 ##
04:45
05:45 ####
11:45
17:45 ########
```

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class EmeltInfo2020maj {
        public static void main(String[] args) throws IOException {

                System.out.println("Az 1. feladat megoldása");
                System.out.println("A tavirathu13.txt fájl beolvasása");

                BufferedReader beolvas = new BufferedReader(new FileReader("tavirathu13.txt"));

                String[] telep = new String[500];
                String[] ido = new String[500];
                String[] szel = new String[500];
                int[] hom = new int[500];

                int db = 0;
                String sor;
                String[] daraboltSor;

                while ((sor = beolvas.readLine()) != null) {
                        daraboltSor = sor.split(" ");
                        db++;
                        telep[db - 1] = daraboltSor[0];
                        ido[db - 1] = daraboltSor[1];
                        szel[db - 1] = daraboltSor[2];
                        hom[db - 1] = Integer.parseInt(daraboltSor[3]);
                }

                beolvas.close();

                System.out.println("\nA 2. feladat megoldása");

                Scanner sc = new Scanner(System.in);
                System.out.print("Adja meg egy település kódját! Település: ");
                String aktTelep = sc.nextLine();

                int i = db;
                while (i >= 1 && !(telep[i - 1].equals(aktTelep))) {
                        i--;
                }

                if (i >= 1) {
                        System.out.println("Az utolsó mérési adat a megadott településről "
                                        + ido[i - 1].substring(0, 2) + ":"
                                        + ido[i - 1].substring(2, 4) + "-kor érkezett.");
                } else {
                        System.out.println("A városból nem érkezett mérési adat.");
                }
                sc.close();
```

```
System.out.println("\nA 3. feladat megoldása");

int min = 1;
int max = 1;
for (i = 2; i <= db; i++) {
        if (hom[i - 1] < hom[min - 1]) {
                min = i;
        }
        if (hom[i - 1] > hom[max - 1]) {
                max = i;
        }
}

System.out.println("A legalacsonyabb hőmérséklet: " + telep[min - 1]
                    + " " + ido[min - 1].substring(0, 2) + ":"
                + ido[min - 1].substring(2, 4) + " " + hom[min - 1] + " fok.");
System.out.println("A legmagasabb hőmérséklet: " + telep[max - 1]
                    + " " + ido[max - 1].substring(0, 2) + ":"
                + ido[max - 1].substring(2, 4) + " " + hom[max - 1] + " fok.");

System.out.println("\nA 4. feladat megoldása");

int szcsdb = 0;

for (i = 1; i <= db; i++) {
        if (szel[i - 1].equals("00000")) {
                szcsdb++;
                System.out.println(telep[i - 1] + " " + ido[i - 1].substring(0, 2) + ":"
                                + ido[i - 1].substring(2, 4));
        }
}
if (szcsdb == 0) {
        System.out.println("Nem volt szélcsend a mérések idején.");
}

System.out.println("\nAz 5. feladat megoldása");

// Településlista készítése
String[] teleplista = new String[db];
int tdb = 0;
int j;
String akttelep;
for (i = 1; i <= db; i++) {
        akttelep = telep[i - 1];
        j = 1;
        while (j <= tdb && !(akttelep.equals(teleplista[j - 1]))) {
                j++;
        }
        if (j > tdb) {
                tdb++;
                teleplista[tdb - 1] = akttelep;
        }
}

int h1, h7, h13, h19;
int homossz;
double homatlag;
for (i = 1; i <= tdb; i++) {
        akttelep = teleplista[i - 1];
        h1 = 0;
        h7 = 0;
        h13 = 0;
        h19 = 0;
        homossz = 0;
        min = 1;
        max = 1;
```

```
                        for (j = 1; j <= db; j++) {
                                if (telep[j - 1].equals(akttelep)) {
                                        if (ido[j - 1].substring(0, 2).equals("01")) {
                                                h1++;
                                                homossz += hom[j - 1];
                                        }
                                        if (ido[j - 1].substring(0, 2).equals("07")) {
                                                h7++;
                                                homossz += hom[j - 1];
                                        }
                                        if (ido[j - 1].substring(0, 2).equals("13")) {
                                                h13++;
                                                homossz += hom[j - 1];
                                        }
                                        if (ido[j - 1].substring(0, 2).equals("19")) {
                                                h19++;
                                                homossz += hom[j - 1];
                                        }

                                        if (hom[j - 1] < hom[min - 1]) {
                                                min = j;
                                        }
                                        if (hom[j - 1] > hom[max - 1]) {
                                                max = j;
                                        }
                                }
                        }

                        System.out.print(akttelep + " ");
                        homatlag = (double) homossz / (h1 + h7 + h13 + h19);

                        if (h1 * h7 * h13 * h19 == 0) {
                                System.out.print("NA");
                        } else {
                                System.out.print("Középhőmérséklet: ");
                                System.out.printf("%.0f", homatlag);
                        }
                        System.out.println("; Hőmérséklet-ingadozás: " + (hom[max - 1] - hom[min - 1]));
                }

                System.out.println("\nA 6. feladat megoldása");

                int n, k;
                for (i = 1; i <= tdb; i++) {
                        akttelep = telep[i - 1];
                        String cim = "" + akttelep + ".txt";
                        PrintWriter kiir = new PrintWriter(new FileWriter(cim));
                        kiir.println(akttelep);

                        for (j = 1; j <= db; j++) {
                                if (telep[j - 1].equals(akttelep)) {
                                        kiir.print(ido[j - 1].substring(0, 2) + ":" + ido[j - 1].substring(2, 4) + " ");
                                        n = Integer.parseInt(szel[j - 1].substring(3));
                                        for (k = 1; k <= n; k++) {
                                                kiir.print("#");
                                        }
                                        kiir.println("");
                                }
                        }

                        kiir.close();
                }
                System.out.println("A fájlok elkészültek.");

        }
}
```

```
Az 1. feladat megoldása
A tavirathu13.txt fájl beolvasása

A 2. feladat megoldása
Adja meg egy település kódját! Település: SM
Az utolsó mérési adat a megadott településről 23:45-kor érkezett.

A 3. feladat megoldása
A legalacsonyabb hőmérséklet: SM 23:45 16 fok.
A legmagasabb hőmérséklet: DC 13:15 35 fok.

A 4. feladat megoldása
BP 01:00
DC 02:15
SN 03:15
BC 04:45
DC 04:45
SN 05:15
SN 05:45
KE 08:45
BC 11:45

Az 5. feladat megoldása
BP Középhőmérséklet: 23; Hőmérséklet-ingadozás: 8
DC Középhőmérséklet: 29; Hőmérséklet-ingadozás: 15
SM Középhőmérséklet: 22; Hőmérséklet-ingadozás: 8
PA Középhőmérséklet: 21; Hőmérséklet-ingadozás: 7
SN Középhőmérséklet: 26; Hőmérséklet-ingadozás: 13
PR Középhőmérséklet: 21; Hőmérséklet-ingadozás: 8
BC NA; Hőmérséklet-ingadozás: 14
PP NA; Hőmérséklet-ingadozás: 6
KE NA; Hőmérséklet-ingadozás: 13

A 6. feladat megoldása
A fájlok elkészültek.


A BC.txt szövegfile:

BC
00:45 ###
01:45 ####
02:45 ######
03:45 ##
04:45
05:45 ####
11:45
17:45 ########
```

2020 május idegen nyelvű: Menetrend

Az ország keleti felében évekkel ezelőtt bevezették az ütemes menetrendet. Ez azt jelenti, hogy a végállomásról minden órában ugyanakkor indulnak a vonatok és menetrend szerint minden állomásra ugyanakkor érkeznek. A jól tervezhető utazás miatt nőtt az utazók száma.

A *vonat.txt* fájlban rögzítették a Szeged-Budapest vonal néhány vonatának indulási és érkezési adatait. A fájl soraiban öt, tabulátorral elválasztott érték található, négy egész szám és egy karakter. Az első szám a vonatazonosító, a második az állomásazonosító, a harmadik és negyedik egy időpont órája és perce. A karakter pedig azt jelzi, hogy a vonat az adott állomásra érkezik (E) vagy éppen indul (I) a megadott időben.

A sorok száma legfeljebb 1000, a vonatok és az állomások azonosítója pedig egy 0 és 20 közötti egész szám. Az óra értéke 0 és 23, a perc 0 és 59 közötti érték. Az állomások 0-tól távolság, a vonatok 1-től indulási idő szerint növekvően sorszámozottak, minden értéket felvesznek.

A fájl a vonatok tényleges útját rögzíti. Az adatok időrendben szerepelnek, azon belül pedig – az induló állomás kivételével – az érkezés mindig megelőzi az indulást. Tudjuk, hogy minden vonat a 0. állomásról indul, és eléri a végállomást, közben minden állomáson megáll, és egyik vonat sem előzi meg a másikat.

Például:

```
…
2 0 6 45 I
1 4 6 49 E
1 4 6 50 I
2 1 6 58 E
1 5 7 0 E
```

Az első sorból leolvasható, hogy a 2. vonat a kiinduló állomásról 6 óra 45 perckor indul. A következő sorban pedig az szerepel, hogy az 1. vonat 6 óra 49 perckor érkezik a 4. állomásra.

Készítsen programot, amely a `vonat.txt` állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `menetrend` néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: 5. feladat)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el a *vonat.txt* fájl tartalmát!
2. Írja a képernyőre a fájlban tárolt vonatok és állomások darabszámát – a kezdő és végállomást is beleértve!
3. Határozza meg, hogy melyik állomáson állt legtöbbet vonat! Adja meg a vonat és az állomás azonosítóját, valamint az állás idejét! Ha több ilyen volt, elég csak az egyiket megadnia.
4. Olvassa be egy vonat azonosítóját, valamint egy időpont óra és perc értékét! A későbbi feladatokban használja ezeket!
5. Ezen a vonalon az előírt menetidő 2 óra 22 perc. Írja a képernyőre, hogy a beolvasott azonosítójú vonat hány perccel tért el ettől! Például: „`A(z) 5. vonat útja 2 perccel rövidebb volt az előírtnál.`”, „`A(z) 5. vonat útja pontosan az előírt ideig tartott.`” vagy „`A(z) 5. vonat útja 3 perccel hosszabb volt az előírtnál.`”
6. Írja a *haladX.txt* fájlba, hogy a beolvasott azonosítójú vonat melyik állomásra mikor érkezett! A fájlnévben az X helyére a beolvasott vonatazonosító kerüljön!

7. Adja meg, hogy a beolvasott időpontban úton lévő, azaz a már elindult, de a végállomást még el nem érő vonatok közül melyik hol tartott! A tesztelés során a következő időpontokra érdemes figyelni: 6:50, 8:45, 9:05, 10:04, 10:20.

Minta a szöveges kimenetek kialakításához:

```
2. feladat
Az állomások száma: 11
A vonatok száma: 12
3. feladat
A(z) 5. vonat a(z) 6. állomáson 10 percet állt.
4. feladat
Adja meg egy vonat azonosítóját! 2
Adjon meg egy időpontot (óra perc)! 7 16
5. feladat
A(z) 2. vonat útja 2 perccel hosszabb volt az előírtnál.
7. feladat
A(z) 1. vonat a 6. állomáson állt.
A(z) 2. vonat a 2. és a 3. állomás között járt.
```

A `halad2.txt` **fájl tartalma:**

```
1. állomás: 6:58
2. állomás: 7:11
3. állomás: 7:31
4. állomás: 7:48
5. állomás: 7:59
6. állomás: 8:11
7. állomás: 8:45
8. állomás: 8:51
9. állomás: 9:0
10. állomás: 9:9
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class EmeltInfo2020mid {
	public static void main(String[] args) throws IOException {

		System.out.println("Az 1. feladat megoldása");
		System.out.println("A vonat.txt fájl beolvasása");

		BufferedReader beolvas = new BufferedReader(new FileReader("vonat.txt"));

		int[] von = new int[1000];
		int[] all = new int[1000];
		int[] ora = new int[1000];
		int[] perc = new int[1000];
		String[] ie = new String[1000];

		int db = 0;
		String sor;
		String[] daraboltSor;

		while ((sor = beolvas.readLine()) != null) {
			daraboltSor = sor.split("\t");
			db++;
			von[db - 1] = Integer.parseInt(daraboltSor[0]);
			all[db - 1] = Integer.parseInt(daraboltSor[1]);
			ora[db - 1] = Integer.parseInt(daraboltSor[2]);
			perc[db - 1] = Integer.parseInt(daraboltSor[3]);
			ie[db - 1] = daraboltSor[4];
		}

		beolvas.close();

		System.out.println("\nA 2. feladat megoldása");

		// A indulási állomás a 0, az utolsó vonat ér be utoljára a végállomásra

		int vondb = von[db - 1];
		int vegall = all[db - 1];
		System.out.println("Az állomások száma: " + (vegall + 1));
		System.out.println("A vonatok száma: " + vondb);

		System.out.println("\nA 3. feladat megoldása");

		int akterk = -1;
		int aktvar = -1;
		int maxvar = -1;
		int maxvon = -1;
		int maxall = -1;
		int i, j, k;
```

```
for (i = 1; i <= vondb; i++) {
        for (j = 1; j <= vegall; j++) {
                for (k = 1; k <= db; k++) {
                        if (von[k - 1] == i && all[k - 1] == j && ie[k - 1].equals("E")) {
                                akterk = 60 * ora[k - 1] + perc[k - 1];
                        }
                        if (von[k - 1] == i && all[k - 1] == j && ie[k - 1].equals("I")) {
                                aktvar = 60 * ora[k - 1] + perc[k - 1] - akterk;
                                if (aktvar > maxvar) {
                                        maxvon = i;
                                        maxall = j;
                                        maxvar = aktvar;
                                }
                        }
                }
        }
}

System.out.println("A(z) " + maxvon + ". vonat a(z) " + maxall + ". állomáson "
                    + maxvar + " percet állt.");

System.out.println("\nA 4. feladat megoldása");

Scanner sc1 = new Scanner(System.in);
System.out.print("Adja meg egy vonat azonosítóját! ");
int aktvon = sc1.nextInt();

Scanner sc2 = new Scanner(System.in);
System.out.print("Adjon meg egy időpontot (óra perc)! ");
String aktido = sc2.nextLine();
int aktora = Integer.parseInt(aktido.split(" ")[0]);
int aktperc = Integer.parseInt(aktido.split(" ")[1]);

sc1.close();
sc2.close();

System.out.println("\nAz 5. feladat megoldása");

int menetido = 60 * 2 + 22;
int aktind = 0;
akterk = 0;
int aktmenetido;
for (i = 1; i <= db; i++) {
        if (von[i - 1] == aktvon && all[i - 1] == 0) {
                aktind = 60 * ora[i - 1] + perc[i - 1];
        }
        if (von[i - 1] == aktvon && all[i - 1] == vegall) {
                akterk = 60 * ora[i - 1] + perc[i - 1];
        }
}
aktmenetido = akterk - aktind;
if (menetido > 0) {
        if (aktmenetido < menetido) {
                System.out.println("A(z) " + aktvon + ". vonat útja " + (menetido - aktmenetido)
                                + " perccel rövidebb volt az előírtnál.");
        } else if (aktmenetido > menetido) {
                System.out.println("A(z) " + aktvon + ". vonat útja " + (aktmenetido - menetido)
                                + " perccel hosszabb volt az előírtnál.");
        } else {
                System.out.println("A(z) " + aktvon + ". vonat útja pontosan az előírt ideig tartott.");
        }
} else {
        System.out.println("Nem volt ilyen vonat.");
}
```

```
                System.out.println("\nA 6. feladat megoldása");

                String cim = "halad" + aktvon + ".txt";
                PrintWriter kiir = new PrintWriter(new FileWriter(cim));

                for (i = 1; i <= db; i++) {
                        if (von[i - 1] == aktvon && ie[i - 1].equals("E")) {
                                kiir.println(all[i - 1] + ". állomás: " + ora[i - 1] + ":" + perc[i - 1]);
                        }
                }

                System.out.println("A halad" + aktvon + ".txt fájl kiírása befejeződött.");
                kiir.close();

                System.out.println("\nA 7. feladat megoldása");

                int aktidoertek = 60 * aktora + aktperc;

                akterk = -1;
                aktind = -1;
                for (i = 1; i <= vondb; i++) {
                        for (j = 1; j <= vegall; j++) {
                                for (k = 1; k <= db; k++) {
                                        if (von[k - 1] == i && all[k - 1] == j - 1 && ie[k - 1].equals("I")) {
                                                aktind = 60 * ora[k - 1] + perc[k - 1];
                                        }
                                        if (von[k - 1] == i && all[k - 1] == j && ie[k - 1].equals("E")) {
                                                akterk = 60 * ora[k - 1] + perc[k - 1];
                                                if (aktind < aktidoertek && akterk > aktidoertek) {
                                                        System.out.println("A(z) " + i + ". vonat a(z) "
                                                                        + (j - 1) + ". és a(z) " + (j)
                                                                +   ". állomások között járt.");
                                                }
                                        }
                                        if (von[k - 1] == i && all[k - 1] == j && ie[k - 1].equals("I")) {
                                                aktind = 60 * ora[k - 1] + perc[k - 1];
                                                if (akterk <= aktidoertek && aktind >= aktidoertek) {
                                                        System.out.println("A(z) " + i + ". vonat a(z) " + j
                                                                        + ". állomáson állt.");
                                                }
                                        }
                                }
                        }
                }

        }
}
```

```
Az 1. feladat megoldása
A vonat.txt fájl beolvasása

A 2. feladat megoldása
Az állomások száma: 11
A vonatok száma: 12

A 3. feladat megoldása
A(z) 5. vonat a(z) 6. állomáson 10 percet állt.

A 4. feladat megoldása
Adja meg egy vonat azonosítóját! 2
Adjon meg egy időpontot (óra perc)! 7 16

Az 5. feladat megoldása
A(z) 2. vonat útja 2 perccel hosszabb volt az előírtnál.

A 6. feladat megoldása
A halad2.txt fájl kiírása befejeződött.

A 7. feladat megoldása
A(z) 1. vonat a(z) 6. állomáson állt.
A(z) 2. vonat a(z) 2. és a(z) 3. állomások között járt.


A halad2.txt szövegfile:

1. állomás: 6:58
2. állomás: 7:11
3. állomás: 7:31
4. állomás: 7:48
5. állomás: 7:59
6. állomás: 8:11
7. állomás: 8:45
8. állomás: 8:51
9. állomás: 9:0
10. állomás: 9:9
```

2020 október: Sorozatok

Sok olyan sorozatrajongó van, aki folyamatosan követi a kedvelt sorozatait. Egy, az angol nyelvű sorozatokért rajongó személy feljegyzést készített egy nyolc hónapos időszak kedvenc sorozatairól.

Például:

```
…
2018.01.19
Puzzles
3x10
43
0
NI
Puzzles
3x11
43
0
…
```

A *lista.txt* fájl a rajongó által kedvelt sorozatok adásba kerülésének dátumát, a sorozat angol címét, az évadot és az epizód számát, az epizód hosszát percben és egy jelzést tartalmaz, hogy a lista készítője megnézte-e már azt az epizódot. Ezek az adatok egymás után külön sorokban szerepelnek. A fájlban biztosan 400-nál kevesebb epizódról van adat, epizódonként 5 sorban. A példában látható, hogy a Puzzles című sorozat 3. évadának 10. epizódja 2018. 01. 19-én került adásba. Az epizód 43 perces, és még nem nézte meg a lista készítője.

- A dátumokat mindig „éééé.hh.nn” formátumban rögzítették. Vannak olyan sorozatrészek, amelyeknek a lista rögzítésekor még nem tudták az adásba kerülésük idejét. Ezeknél a dátumhelyett mindig az „***NI***” rövidítés szerepel.
- Az évad jelzése vezető nullák nélkül történik, az epizód számát pedig mindig két számjeggyel rögzítették. Az évad és az epizód számát egy „***x***” választja el egymástól.
- Az egyes sorozatok epizódjai mindig ugyanolyan hosszúak.
- Az epizóddal kapcsolatos utolsó adat értéke „***0***” vagy „***1***”. Az 1-es számjegy jelöli, hogy az adott részt már megtekintette a lista készítője, a 0 pedig azt, hogy még nem látta.

Készítsen programot a *lista.txt* állomány adatainak feldolgozására! A program forráskódját mentse *sorozatok* néven! (A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.)

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például `2. feladat:`)! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott.

1. Olvassa be és tárolja el a *lista.txt* fájl tartalmát!
2. Írassa ki a képernyőre, hogy hány olyan epizód adatait tartalmazza a fájl, amelynek ismert az adásba kerülési dátuma!
3. Határozza meg, hogy a fájlban lévő epizódok hány százalékát látta már a listát rögzítő személy! A százalékértéket a minta szerint, két tizedesjeggyel jelenítse meg a képernyőn!
4. Számítsa ki, hogy összesen mennyi időt töltött a személy az epizódok megnézésével! Az eredményt a minta szerint nap, óra, perc formában adja meg!
5. Kérjen be a felhasználótól egy dátumot „éééé.hh.nn” formában! Határozza meg, hogy az adott dátumig megjelent epizódokból melyeket nem látta még! Az aznapi epizódokat is számolja bele! A feltételnek megfelelő epizódok esetén írja a képernyőre tabulátorral elválasztva az évad- és az epizódszámot, valamint a sorozat címét a minta szerint!
6. Készítse el az alábbi algoritmus alapján a hét napját meghatározó függvényt! A függvény neve *Hetnapja* legyen! A függvény az év, hónap és nap megadása után szöveges eredményként visszaadja, hogy az adott nap a hét melyik napja volt. (Az `a` és `b` egész számok maradékos osztása esetén az `a div b` kifejezés adja meg a hányadost, az `a mod b` pedig a maradékot, például `17 div 7 = 2` és `17 mod 7 = 3`.)

```
Függvény hetnapja(ev, ho, nap : Egész) : Szöveg
  napok: Tömb(0..6: Szöveg)= ("v", "h", "k", "sze", "cs", "p", "szo")
  honapok: Tömb(0..11: Egész)= (0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4)
  Ha ho < 3 akkor ev := ev -1
  hetnapja := napok[(ev + ev div 4 – ev div 100 +
  ev div 400 + honapok[ho-1] + nap) mod 7]
Függvény vége
```

7. Kérjen be a felhasználótól egy napot az előző feladatban látható rövidített alakban! A napokat egy (h, k, p, v), kettő (cs), vagy három (sze, szo) karakterrel adja meg! Határozza meg, hogy a fájlban lévő sorozatok közül melyike(ke)t vetítik az adott napon! A sorozatok nevét a minta szerint jelenítse meg a képernyőn! Ha az adott napon egy sorozatot sem adtak adásba, akkor „Az adott napon nem kerül adásba sorozat.” üzenetet jelenítse meg!
8. Határozza meg sorozatonként az epizódok összesített vetítési idejét és az epizódok számát! A számításnál vegye figyelembe a vetítési dátummal nem rendelkező epizódokat is! A megoldás során felhasználhatja, hogy egy sorozat epizódjainak adatai egymást követik a forrásállományban. A listát írja ki a *summa.txt* fájlba! A fájl egy sorában a sorozat címe, az adott sorozatra vonatkozó összesített vetítési idő percben és az epizódok száma szerepeljen szóközzel elválasztva!

Minta a szöveges kimenetek kialakításához:

```
2. feladat
A listában 202 db vetítési dátummal rendelkező epizód van.
3. feladat
A listában lévő epizódok 45,66%-át látta.
4. feladat
Sorozatnézéssel 2 napot 15 órát és 32 percet töltött.
5. feladat
Adjon meg egy dátumot! Dátum= 2017.10.18
7x01 The Fable
7x02 The Fable
15x04 Military Police
5x03 Spy School
5x04 Spy School
4x04 The Elite Minds
7. feladat
Adja meg a hét egy napját (például cs)! Nap= cs
The Hospital
Spectacular Power
Upper Story
Chicago Flame
Shrinktime
```

Minta a *summa.txt* fájl kialakításához:

```
Games 420 7
The Fable 588 14
The IT Guy 450 10
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Sorozatok
 *
 * @author Klemand
 */

public class EmeltInfo2020okt {

	public static void main(String[] args) throws IOException {

		System.out.println("A 1. feladat megoldása");

		BufferedReader beolvas = new BufferedReader(new FileReader("lista.txt"));

		String[] datum = new String[400];
		String[] cim = new String[400];
		String[] evadep = new String[400];
		int[] hossz = new int[400];
		int[] jel = new int[400];

		int db = 0;
		String sor;

		while ((sor = beolvas.readLine()) != null) {
			db++;
			datum[db - 1] = sor;
			cim[db - 1] = beolvas.readLine();
			evadep[db - 1] = beolvas.readLine();
			hossz[db - 1] = Integer.parseInt(beolvas.readLine());
			jel[db - 1] = Integer.parseInt(beolvas.readLine());

		}
		beolvas.close();
		System.out.println("A lista.txt fájl beolvasása megtörtént.");

		System.out.println("\nA 2. feladat megoldása ");

		int ism = db;
		int i;
		for (i = 1; i <= db; i++) {
			if (datum[i - 1].equals("NI")) {
				ism--;
			}
		}
		System.out.println("A listában " + ism + " db vetítési dátummal rendelkező epizód van.");

		System.out.println("\nA 3. feladat megoldása ");

		int latott = 0;

		for (i = 1; i <= db; i++) {
			if (jel[i - 1] == 1) {
				latott++;
			}
		}
		double szazalek;
		szazalek = (100 * (double) latott / db);
		System.out.printf("A listában lévő epizódok %.2f ", szazalek);
		System.out.println("%-át látta.");
```

```
System.out.println("\nA 4. feladat megoldása ");

int ido = 0;

for (i = 1; i <= db; i++) {
        if (jel[i - 1] == 1) {
                ido += hossz[i - 1];
        }
}

int nap = 0;
int ora = 0;
int perc = 0;

perc = ido % 60;
ora = ido / 60;
nap = ora / 24;
ora = ora % 24;

System.out.println("Sorozatnézéssel " + nap + " napot " + ora + " órát és " + perc +
                   " percet töltött.");

System.out.println("\nAz 5. feladat megoldása ");

Scanner sc = new Scanner(System.in);
System.out.print("Adjon meg egy dátumot! Dátum= ");
String megadottDatum = sc.nextLine();

System.out.println("A megadott dátumig megjelent epizódokból a következőket nem látta még:\n");

for (i = 1; i <= db; i++) {
        if (datum[i - 1].compareTo(megadottDatum) <= 0) {
                if (jel[i - 1] == 0) {
                        System.out.println(evadep[i - 1] + "\t" + cim[i - 1]);
                }
        }
}

System.out.println("\nA 6. feladat megoldása");
System.out.println("A hetnapja függvény elkészült.");
```

```
System.out.println("\nA 7. feladat megoldása ");

System.out.print("Adja meg a hét egy napját (például cs)! Nap= ");
String megadottNap = sc.nextLine();
int aktEv;
int aktHo;
int aktNap;

String[] filmLista = new String[400];
String aktFilm;

int filmListaDb = 0;
int j;



for (i = 1; i <= db; i++) {

        if (!(datum[i - 1].equals("NI"))) {
                aktEv = Integer.parseInt(datum[i - 1].substring(0, 4));
                aktHo = Integer.parseInt(datum[i - 1].substring(5, 7));
                aktNap = Integer.parseInt(datum[i - 1].substring(8));

                if (hetnapja(aktEv, aktHo, aktNap).equals(megadottNap)) {
                        aktFilm = cim[i - 1];
                        j = 1;
                        while (j <= filmListaDb && !(filmLista[j - 1].equals(aktFilm))) {
                                j++;
                        }
                        if (j > filmListaDb) {
                                filmListaDb++;
                                filmLista[filmListaDb - 1] = aktFilm;
                        }
                }
        }
}

if (filmListaDb > 0) {
        System.out.println("A következő sorozatokat vetítik ezen a napon:\n");
        for (i = 1; i <= filmListaDb; i++) {
                System.out.println(filmLista[i - 1]);
        }
} else {
        System.out.println("Az adott napon nem kerül adásba sorozat.");
}

sc.close();
```

```
		System.out.println("\nA 8. feladat megoldása ");

		String[] osszFilmLista = new String[400];
		int osszFilmListaDb = 0;
		int[] osszIdo = new int[400];
		int[] osszEpizod = new int[400];

		for (i = 1; i <= db; i++) {
			aktFilm = cim[i - 1];

			j = 1;
			while (j <= osszFilmListaDb && !(osszFilmLista[j - 1].equals(aktFilm))) {
				j++;
			}
			if (j <= osszFilmListaDb) {
				osszIdo[j - 1] += hossz[i - 1];
				osszEpizod[j - 1]++;

			} else {
				osszFilmListaDb++;
				osszFilmLista[osszFilmListaDb - 1] = aktFilm;
				osszIdo[osszFilmListaDb - 1] = hossz[i - 1];
				osszEpizod[osszFilmListaDb - 1] = 1;
			}
		}



		PrintWriter kiir = new PrintWriter(new FileWriter("summa.txt"));



		for (i = 1; i <= osszFilmListaDb; i++) {
			kiir.println(osszFilmLista[i - 1] + " " + osszIdo[i - 1] + " " + osszEpizod[i - 1]);
		}

		kiir.close();
		System.out.println("A summa.txt fájl kiírása befejeződött.");

	}

	public static String hetnapja(int ev, int ho, int nap) {
		String[] napok = { "v", "h", "k", "sze", "cs", "p", "szo" };
		int[] honapok = { 0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4 };
		if (ho < 3) {
			ev--;
		}
		return napok[(ev + ev / 4 - ev / 100 + ev / 400 + honapok[ho - 1] + nap) % 7];
	}

}
```

```
A 1. feladat megoldása
A lista.txt fájl beolvasása megtörtént.

A 2. feladat megoldása
A listában 202 db vetítési dátummal rendelkező epizód van.

A 3. feladat megoldása
A listában lévő epizódok 45,66 %-át látta.

A 4. feladat megoldása
Sorozatnézéssel 2 napot 15 órát és 32 percet töltött.

Az 5. feladat megoldása
Adjon meg egy dátumot! Dátum= 2017.10.18
A megadott dátumig megjelent epizódokból a következőket nem látta még:

7x01    The Fable
7x02    The Fable
15x04   Military Police
5x03    Spy School
5x04    Spy School
4x04    The Elite Minds

A 6. feladat megoldása
A hetnapja függvény elkészült.

A 7. feladat megoldása
Adja meg a hét egy napját (például cs)! Nap= cs
A következő sorozatokat vetítik ezen a napon:

The Hospital
Spectacular Power
Upper Story
Chicago Flame
Shrinktime

A 8. feladat megoldása
A summa.txt fájl kiírása befejeződött.


A summa.txt szövegfile:

Games 420 7
The Fable 588 14
The IT Guy 450 10
The Hospital 378 9
The Killers 504 12
Maniac Man 630 15
Adventures with Eric 450 10
Military Police 480 12
The Streets of LA 520 13
Puzzles 688 16
Spy School 495 11
Spectacular Power 588 14
Upper Story 286 13
Chicago Flame 462 11
The Elite Minds 672 16
Shrinktime 945 35
Perfectly Numb 60 1
```

Előzmény: Klasszikus programozás Java nyelven I. Bevezető gyakorlatok

Klasszikus programozás

Java nyelven

I.

Bevezető gyakorlatok

Eclipse (Neon, Oxygen)

Klement András

2016 – 2018