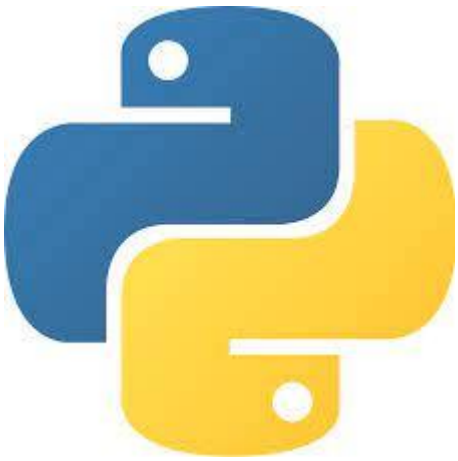
Az emelt szintű

informatika érettségi

programozási feladatainak megoldása

Python nyelven



Klement András

2022-23

Utolsó módosítás: 2023. 10. 28.

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.

```
"""
2003. minta 1: Triatlon
@author Klemand66
"""

print("1. feladat\n")

betxt = open("triatlon.be")
eredmenyek=[] # a versenyzők eredményeinek listája

n=int(betxt.readline().strip()) # a versenyzők száma

for i in range(n):
    nev=betxt.readline().strip()            # név
    uszas=int(betxt.readline().strip())     #úszás
    kp=int(betxt.readline().strip())        #kerékpározás
    futas=int(betxt.readline().strip())     #futás
    osszido=uszas+kp+futas                  #összidő
    eredmeny=[nev,uszas,kp,futas,osszido]   # egy versenyző eredményei
    # indexek: 0    1    2   3      4
    eredmenyek.append(eredmeny)

betxt.close()
print("Az adatok beolvasása megtörtént.\n")


print("2. feladat\n")

# Rendezzük az eredmenyek listát összidő szerint!

eredmenyek.sort(key=lambda eredmeny: eredmeny[4])

print("Az első három helyezett az összidők alapján: \n")

for i in range(3):
    print(f' {i+1}. {eredmenyek[i][0]}')
print()


print("3. feladat\n")

vu=3.6*1500/eredmenyek[0][1]
vk=3.6*40000/eredmenyek[0][2]
vf=3.6*10000/eredmenyek[0][3]

print(f' Az első helyezett neve {eredmenyek[0][0]}, átlagsebessége:')
print(f' úszásban {vu:.02f} km/h, kerékpározásban {vk:.2f} km/h, \
futásban {vf:.2f} km/h.\n')


print("4. feladat\n")

def konvertalo(t):
    mp=t%60
    t=t//60
    perc=t%60
    ora=t//60
    return f'{ora:02}:{perc:02}:{mp:02}' # bevezető nullák

for eredmeny in eredmenyek:
    eredmeny[4]=konvertalo(eredmeny[4])

print("A konvertálás megtörtént.\n")
```

```
print("5. és 6. feladat\n")

kitxt = open("triatlon.ki","w")

for eredmeny in eredmenyek:
    kitxt.write(f' {eredmeny[0]} {eredmeny[4]}\n')

kitxt.close()
print("A triatlon.ki fájl kiíratása befejeződött.\n")


print("7. feladat\n")

kitxt = open("reszer.ki","w")

for i in range(1,4):
    eredmenyek.sort(key=lambda eredmeny: eredmeny[i])
    kitxt.write(f' {eredmenyek[0][0]} {konvertalo(eredmenyek[0][i])}\n')

kitxt.close()
print("A reszer.ki fájl kiíratása befejeződött.\n")

input("A befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2003mt1/EmeltInfo2003mt1.py ==================
1. feladat

Az adatok beolvasása megtörtént.

2. feladat

Az első három helyezett az összidők alapján:

 1. Pál Péter
 2. Kiss Pál
 3. Tóth Pál

3. feladat

 Az első helyezett neve Pál Péter, átlagsebessége:
 úszásban 3.75 km/h, kerékpározásban 28.82 km/h, futásban 12.20 km/h.

4. feladat

A konvertálás megtörtént.

5. és 6. feladat

A triatlon.ki fájl kiíratása befejeződött.

7. feladat

A reszer.ki fájl kiíratása befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!


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!)

```
"""
2003. minta 2: Kugli
@author Klemand66
"""

print("1. feladat\n")

eredm=["eredm1.txt","eredm2.txt","eredm3.txt","eredm4.txt"]
korok=[]
nevek=[]
pontok=[]

betxt = open(eredm[0])

for i in range(5):
    korok.append(1)
    nevek.append(betxt.readline().strip())
    pontok.append(int(betxt.readline().strip()))

betxt.close()
print("Az eredm1.txt fájl adatainak beolvasása megtörtént.\n")


print("2. feladat\n")

print("Az 1. körben hibapontot kaptak:\n")

for i in range(1,5):
    if pontok[i]<pontok[i-1]:
        print(f' {nevek[i]}')
print()


print("3. feladat\n")

for k in range(1,4):
    betxt = open(eredm[k])
    for i in range(5):
        korok.append(k+1)
        nevek.append(betxt.readline().strip())
        pontok.append(int(betxt.readline().strip()))
    betxt.close()

print("Az összes fájl adatainak beolvasása megtörtént.\n")


print("4. feladat\n")

print("A ledöntött bábuk száma:\n")

nevsor=[]
osszpontok=[]

for i,nev in enumerate(nevek):
    if nev in nevsor:
        if pontok[i]!=10:
            osszpontok[nevsor.index(nev)]+=pontok[i]
    else:
        nevsor.append(nev)
        osszpontok.append(pontok[i]) # először még biztosan nem esett ki

szelessegek=list(map(lambda nev:len(nev),nevsor))
szel=max(szelessegek)

for i,nev in enumerate(nevsor):
    print(f' {nev:{szel}}  {osszpontok[i]}')
print()


print("5. feladat\n")

maxpont=max(osszpontok)
print(f'A legtöbb pontot elért versenyző: {nevsor[osszpontok.index(maxpont)]}\
 {maxpont} pont\n');
```

```
print("6. feladat\n")

print("A kiesett versenyzők és a kiesés köre:\n")

kiesettek=[]
kieseskorei=[]

for i,pont in enumerate(pontok):
    if pont==10:
        if nevek[i] not in kiesettek:
            kiesettek.append(nevek[i])
            kieseskorei.append(korok[i]-1)


for i,nev in enumerate(kiesettek):
    print(f' {kiesettek[i]} {kieseskorei[i]}')
print()


print("7. feladat\n")

print("A 9 pontos versenyzők és a 9 pontos körök:\n")

kilencpontosak=[]
kilencpontoskorok=[]

for i,pont in enumerate(pontok):
    if pont==9:
        nev=nevek[i]
        kor=korok[i]
        if nev in kilencpontosak:
            kilencpontoskorok[kilencpontosak.index(nev)].append(kor)
        else:
            kilencpontosak.append(nev)
            kilencpontoskorok.append([kor])

for i,nev in enumerate(kilencpontosak):
    print(f' {kilencpontosak[i]} ',end="")
    for kor in kilencpontoskorok[i]:
        print(f'{kor} ',end="")
    print()
print()

input("A befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2003mt2/EmeltInfo2003mt2.py ==================
1. feladat

Az eredm1.txt fájl adatainak beolvasása megtörtént.

2. feladat

Az 1. körben hibapontot kaptak:

 Kiss Géza
 Kovács Éva

3. feladat

Az összes fájl adatainak beolvasása megtörtént.

4. feladat

A ledöntött bábuk száma:

 Gipsz Jakab  20
 Kelep Elek   32
 Kiss Géza    12
 Nagy Péter   33
 Kovács Éva   14

5. feladat

A legtöbb pontot elért versenyző: Nagy Péter 33 pont

6. feladat

A kiesett versenyzők és a kiesés köre:

 Kiss Géza 2
 Kovács Éva 2
 Gipsz Jakab 3

7. feladat

A 9 pontos versenyzők és a 9 pontos körök:

 Kelep Elek 1 4
 Nagy Péter 2 4

A befejezéshez nyomd meg az ENTER billentyűt!
```

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!

```
"""
2004. május: Személyazonosító jel
@author Klemand66
"""

from datetime import *


print("\na) feladat\n")

szj1=input("Adja meg egy személyazonosító jel első tíz jegyét! ")
szj1=szj1.strip()



print("\nb) feladat\n")

if szj1[0] in "13":
    print("A megadott személy férfi.")
elif szj1[0] in "24":
    print("A megadott személy nő.")
else:
    print("A megadott személyazonosító jel hibás.")



print("\nc) feladat\n")

print(f'A megadott személy születési sorszáma: {szj1[7:]}')


print("\nd) feladat\n")

def szev(kod):
    ev=int(kod[1:3])
    if kod[0] in "12":
        ev+=1900
    else:
        ev+=2000
    return ev

most=datetime.now()
ideiev=most.year
# vagy egyben ideiev=datetime.now().year

print(f'A megadott személynek idén a(z) {ideiev-szev(szj1)}. születésnapja van.')


print("\ne) feladat\n")

szj2=input("Kérem egy másik személyazonosító jel első tíz jegyét! ").strip()


print("\nf) feladat\n")

datum1=str(szev(szj1))+szj1[3:]
datum2=str(szev(szj2))+szj2[3:]

if datum1<datum2: # sztringek esetén karakterenként hasonlít, ami nekünk éppen jó
    print("Az első személy az idősebb.")
else:
    print("A második személy az idősebb.")


print("\ng) feladat\n")

print(f'A két megadott személy születési éveinek különbsége: {abs(szev(szj2)-szev(szj1))}')
```

```
print("\nh) feladat\n")

s=0

for i in range(9,-1,-1):
    s+=int(szj2[i])*(10-i)

ell=s%11
if ell==10:
    print("Hibás a születési sorszám.")
else:
    print(f'A teljes második kód: {szj2+str(ell)}')
# A valóságban az i. jegyet i-vel szorozzák. A feladat szerint az én azonosítóm hibás lenne.


print("\ni) feladat\n")

kitxt = open("szemszam.txt","w")

kitxt.write(f'{szj1}\n')
kitxt.write(f'{szj2}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

input("A befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2004maj\EmeltInfo2004maj.py ==================

a) feladat

Adja meg egy személyazonosító jel első tíz jegyét! 4010101010

b) feladat

A megadott személy nő.

c) feladat

A megadott személy születési sorszáma: 010

d) feladat

A megadott személynek idén a(z) 21. születésnapja van.

e) feladat

Kérem egy másik személyazonosító jel első tíz jegyét! 1560403100

f) feladat

A második személy az idősebb.

g) feladat

A két megadott személy születési éveinek különbsége: 45

h) feladat

Hibás a születési sorszám.

i) feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A szemszam.txt szövegfájl:

```
4010101010
1560403100
```

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.)

```
"""
2005. május: Lottó
@author Klemand66
"""

print("\n1. feladat\n")

sor=input("Kérem az 52. hét megadott lottószámait (89 24 34 11 64) szóközökkel elválasztva! ")
darsor=sor.strip().split()

"""
A lotto52het lista elkészítése ciklussal:
lotto52het=[]
for elem in darsor:
    lotto52het.append(int(elem))

A map() függvénnyel:
"""
lotto52het=list(map(int,darsor))

print("\n2. feladat\n")

lotto52het.sort()

print("Az 52. hét lottószámai nagyság szerint rendezve: ", end="")
for szam in lotto52het:
    print(szam,end=" ")
print()


print("\n3. feladat\n")

het=int(input("Kérem a vizsgált hét sorszámát 1 és 51 között! "))


print("\n4. feladat\n")

lottok=[]

betxt = open("lottosz.dat")

for i in range(51):
    sor=betxt.readline()
    darsor=sor.strip().split()
    lotto=list(map(int,darsor))
    lottok.append(lotto)

betxt.close()

print(f'A(z) {het}. hét lottószámai : ', end="")
for szam in lottok[het-1]: # az 1. hét számai a lottok[0] listában vannak
    print(szam,end=" ")
print()


print("\n5. feladat\n")

# A lottószámok gyakorisági táblázatának elkészítése

gyak=[]
for i in range(90):
    gyak.append(0) # kezdőértékek

for lotto in lottok:
    for szam in lotto:
        gyak[szam-1]+=1

van=gyak.count(0)>0

if van:
    kinemhuzottak=list(filter(lambda i: gyak[i-1]==0,range(1,91))) # szorgalmi feladat
    print(f'Van. (Összesen {gyak.count(0)} db: {kinemhuzottak})')
else:
    print("Nincs.")
```

```
print("\n6. feladat\n")

"""
A kérdés kétféleképpen is értelmezhető:
a) Az összes kihúzott lottószám közül hány volt páratlan?
b) A heti lottószám húzások közül hány héten volt páratlan szám?
"""

# a)

db=0
for i in range(0,90,2): # csak a páratlan számokat nézzük, pl. gyak[10] a 11 gyakorisága
    db+=gyak[i]

print(f'a) Összesen {db} alkalommal húztak ki páratlan számot.\n')

# b)

db1=0
db2=0
for lotto in lottok:
    paratlanok=list(filter(lambda szam:szam%2==1,lotto))
    if len(paratlanok)>0:
        db1+=1
        db2+=len(paratlanok) # ez az a) feladat eredménye

print(f'b) {db1} héten húztak ki páratlan számot. Összesen {db2} alkalommal.')

print("\n7. feladat\n")

lottok.append(lotto52het)

kitxt = open("lotto52.ki","w")

for lotto in lottok:
    for szam in lotto:
        kitxt.write(f'{szam:2} ') # kicsit szebben
    kitxt.write("\n")

kitxt.close()
print(f'A lotto52.ki fájl kiíratása és a fájl lezárása sikeresen befejeződött.')


print("\n8. feladat\n")

# A gyakorisági táblázatot ki kell egészíteni az 52. hét számaival

for szam in lotto52het:
    gyak[szam-1]+=1

for i,n in enumerate(gyak):
    print(n, end=" ")
    if (i+1)%15==0:
        print()

print()


print("\n9. feladat\n")

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]

print("Egyszer sem kihúzott prímek: ",end="")

for p in primek:
    if gyak[p-1]==0:
        print(p,end=" ")
print("\n")


input("A befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2005maj\EmeltInfo2005maj.py ==================

1. feladat

Kérem az 52. hét megadott lottószámait (89 24 34 11 64) szóközökkel elválasztva!  89 24 34 11 64

2. feladat

Az 52. hét lottószámai nagyság szerint rendezve: 11 24 34 64 89

3. feladat

Kérem a vizsgált hét sorszámát 1 és 51 között! 7

4. feladat

A(z) 7. hét lottószámai : 21 29 37 48 68

5. feladat

Van. (Összesen 4 db: [11, 19, 27, 63])

6. feladat

a) Összesen 126 alkalommal húztak ki páratlan számot.

b) 49 héten húztak ki páratlan számot. Összesen 126 alkalommal.

7. feladat

A lotto52.ki fájl kiíratása és a fájl lezárása sikeresen befejeződött.

8. feladat

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. feladat

Egyszer sem kihúzott prímek: 19

A befejezéshez nyomd meg az ENTER billentyűt!
```

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
```

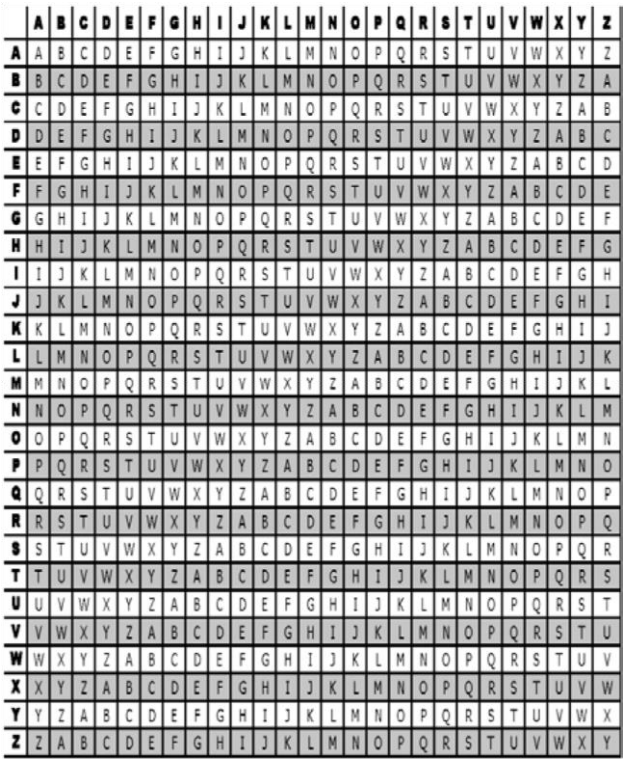
2005. október: Vigenère tábla

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

```
"""
2005. október: Vignère tábla
@author Klemand66
"""

print("\n1. feladat\n")

nysz=input("Kérek egy max. 255 karakterből álló ékezetes szöveget!\n")


print("\n2. feladat\n")

nysz=nysz.upper()

mabc = "AÁBCDEÉFGHIÍJKLMNOÓÖŐPQRSTUÚÜŰVWXYZ"
aabc = "AABCDEEFGHIIJKLMNOOOOPQRSTUUUUVWXYZ"

asz="" # az átalakított szöveg

for k in nysz:
    if k in mabc:
        i=mabc.index(k)
        asz+=aabc[i]

print("A nyílt szöveg átalakítása megtörtént.")


print("\n3. feladat\n")

print(asz)


print("\n4. feladat\n")

kulcs=input("Kérem a kulcsszót, egy max. 5 karakterből álló ékezetmentes szót! ")

kulcs=kulcs.upper()


print("\n5. feladat\n")

n=len(asz)//len(kulcs)
m=len(asz)%len(kulcs)

ksz=n*kulcs+kulcs[:m] # a kulcsszöveg
print(ksz)


print("\n6. feladat\n")

betxt = open("Vtabla.dat")

tabla=[]
for sor in betxt:
    sor=sor.strip()
    tabla.append(list(sor))

betxt.close()

# Vegyük észre, hogy a táblázat első oszlopa és sora is éppen az angol abc.

abc = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

tsz="" # a titkosított szöveg

for i in range(len(asz)):
    s=abc.index(asz[i])
    o=abc.index(ksz[i])
    tsz+=tabla[s][o]

print("A fájlbeolvasás és a kódolás megtörtént.")
```

```
print("\n7. feladat\n")

kitxt = open("kodolt.dat","w")

print(tsz,"\n")
kitxt.write(f'{tsz}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

# Szorgalmi feladat: a kódolt szöveg visszafejtése

vsz="" # a visszafejtett szöveg

for i in range(len(tsz)):
    """
    A tábla oszlopát a kulcsszöveg aktuális betűje határozza meg.
    Ebből az oszlopból kell kiválasztanunk a titkos szöveg aktuális betűjét,
    megkeresni, hogy melyik sorban van, annak az első eleme kell.
    """
    o=abc.index(ksz[i])
    tablaoszlop=list(map(lambda i:tabla[i][o],range(len(tabla))))
    # a kétdimenziós lista oszlopából egy sima listát készítettünk
    s=tablaoszlop.index(tsz[i])
    vsz+=tabla[s][0]

print(f'A kódolt szöveg visszafejtése:\n{vsz}\n')

input("A befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2005okt/EmeltInfo2005okt.py ==================

1. feladat

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

2. feladat

A nyílt szöveg átalakítása megtörtént.

3. feladat

AHUSITOTOFOLEEZUSTHOLDSUGARKUSZIK

4. feladat

Kérem a kulcsszót, egy max. 5 karakterből álló ékezetmentes szót! hold

5. feladat

HOLDHOLDHOLDHOLDHOLDHOLDHOLDHOLDH

6. feladat

A fájlbeolvasás és a kódolás megtörtént.

7. feladat

HVFVPHZWVTZOLSKXZHSRSRDXNOCNBGKLR

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A kódolt szöveg visszafejtése:
AHUSITOTOFOLEEZUSTHOLDSUGARKUSZIK

A befejezéshez nyomd meg az ENTER billentyűt!


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!

```
"""
2006. február: Telefonszámla
@author Klemand66
"""

print("\n1. feladat\n")

def mobil_e(tsz):
    a=tsz[:2]
    return  a=="39" or a=="41" or a=="71"

tszam=input("Kérek egy telefonszámot: ")
tszam=tszam.strip()

if mobil_e(tszam):
    print(f'A megadott {tszam} szám mobilszám.')
else:
    print(f'A megadott {tszam} szám nem mobilszám.')


print("\n2. feladat\n")

def szperc(ki,vi):
    bi=vi-ki # a beszélgetési idő mp-ben
    bperc=bi//60
    bmp=bi%60
    if bmp>0:
        bperc+=1
    return bperc

kezdet=input("Kérem egy hívás kezdetét óra perc másodperc formában: ")
kezdet=kezdet.strip().split()
kora=int(kezdet[0])
kperc=int(kezdet[1])
kmp=int(kezdet[2])
kido=3600*kora+60*kperc+kmp

veg=input("Kérem egy hívás végét óra perc másodperc formában: ")
veg=veg.strip().split()
vora=int(veg[0])
vperc=int(veg[1])
vmp=int(veg[2])
vido=3600*vora+60*vperc+vmp

print(f'A beszélgetés számlázás szempontjából {szperc(kido,vido):2} perces.')


print("\n3. feladat\n")

betxt = open("hivasok.txt")

korak=[] # a csúcsidőhöz csak ez kell
kidok=[]
vidok=[]
tszamok=[]

for sor in betxt:
    darsor=sor.strip().split()
    kora=int(darsor[0])
    korak.append(kora)
    kperc=int(darsor[1])
    kmp=int(darsor[2])
    kido=3600*kora+60*kperc+kmp
    kidok.append(kido)
    vora=int(darsor[3])
    vperc=int(darsor[4])
    vmp=int(darsor[5])
    vido=3600*vora+60*vperc+vmp
    vidok.append(vido)
    sor = next(betxt)
    tszam=sor.strip()
    tszamok.append(tszam)

betxt.close()
```

```
kitxt = open("percek.txt","w")

for i in range(len(kidok)):
    kitxt.write(f'{szperc(kidok[i],vidok[i]):2} {tszamok[i]}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


print("\n4. feladat\n")

def csucsido_e(ko):
    return ko>=7 and ko<18

csi=0
csik=0

for kora in korak:
    if csucsido_e(kora):
        csi+=1
    else:
        csik+=1

print(f'{csi} db hívás történt csúcsidőben és {csik} db csúcsidőn kívül.')


print("\n5. feladat\n")

mperc=0
vperc=0

for i,tsz in enumerate(tszamok):
    szp=szperc(kidok[i],vidok[i])
    if mobil_e(tsz):
        mperc+=szp
    else:
        vperc+=szp

print(f'Mobilszámmal {mperc} percet, vezetékessel {vperc} percet beszélt.')


print("\n6. feladat\n")

csdij=0

for i,kora in enumerate(korak):
    if csucsido_e(kora):
        if mobil_e(tszamok[i]):
            csdij+=69.175*szperc(kidok[i],vidok[i])
        else:
            csdij+=30*szperc(kidok[i],vidok[i])

print(f'A csúcsdíjas hívásokért {csdij:.0f} Ft-ot kell fizetnie.\n')


input("A befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2006feb/EmeltInfo2006feb.py ==================

1. feladat

Kérek egy telefonszámot: 707172737
A megadott 707172737 szám nem mobilszám.

2. feladat

Kérem egy hívás kezdetét óra perc másodperc formában: 17 7 05
Kérem egy hívás végét óra perc másodperc formában: 17 09 6
A beszélgetés számlázás szempontjából  3 perces.

3. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

4. feladat

63 db hívás történt csúcsidőben és 22 db csúcsidőn kívül.

5. feladat

Mobilszámmal 101 percet, vezetékessel 245 percet beszélt.

6. feladat

A csúcsdíjas hívásokért 9709 Ft-ot kell fizetnie.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A percek.txt szöveges fájl

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

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!

```
"""
2006. május: Fehérje
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("aminosav.txt")

asavak=[]

for sor in betxt:
    asav=[sor.strip()]                      # 0: rövidítés
    sor = next(betxt).strip()
    asav.append(sor)                        # 1: betűjel
    sor = next(betxt).strip()
    asav.append(int(sor))                   # 2: C
    sor = next(betxt).strip()
    asav.append(int(sor))                   # 3: H
    sor = next(betxt).strip()
    asav.append(int(sor))                   # 4: O
    sor = next(betxt).strip()
    asav.append(int(sor))                   # 5: N
    sor = next(betxt).strip()
    asav.append(int(sor))                   # 6: S
    asavak.append(asav)

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")


print("\n2. feladat\n")

at=[12,1,16,14,32]

"""
Egy új oszlopot illesztünk a táblázat végére.
"""

for asav in asavak:
    rmt=0
    for i in range(2,7):
        rmt+=asav[i]*at[i-2]
    asav.append(rmt)                        # 7: relatív molekulatömeg

print("A relatív molekulatömegek meghatározása befejeződött.")


kitxt = open("eredmeny.txt","w")

print("\n3. feladat\n")
kitxt.write("3. feladat\n")

asavak.sort(key=lambda asav:asav[7])

for asav in asavak:
    print(f'{asav[0]} {asav[7]}')
    kitxt.write(f'{asav[0]} {asav[7]}\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


betxt = open("bsa.txt")

bsa=[]
for sor in betxt:
        bsa.append(sor.strip())

betxt.close()

kitxt = open("eredmeny.txt","a")
```

```
print("\n4. feladat\n")
kitxt.write("\n4. feladat\n")

jeloszlop=list(map(lambda i:asavak[i][1],range(len(asavak))))

elemjel=["C","H","O","N","S"]
elemdb=[0,0,0,0,0]

for jel in bsa:
    i=jeloszlop.index(jel)
    for j in range(5):
        elemdb[j]+=asavak[i][j+2]

elemdb[1]-=2*(len(bsa)-1) # A vízmolekulák kilépésének beszámítása
elemdb[2]-=(len(bsa)-1)

print(f'A BSA nevű fehérje összegképlete: ',end="")
kitxt.write(f'A BSA nevű fehérje összegképlete: ')

for i,jel in enumerate(elemjel):
    print(f'{jel} {elemdb[i]} ',end="")
    kitxt.write(f'{jel} {elemdb[i]} ')

print()
kitxt.write("\n")

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


print("\n5. feladat\n")

aktkezdet=None
akthossz=0
maxkezdet=None
maxhossz=0

for i,jel in enumerate(bsa):
    if aktkezdet==None:
        aktkezdet=i
    akthossz+=1
    if akthossz>maxhossz:
        maxkezdet=aktkezdet
        maxhossz=akthossz
    if jel in "YWF": # A hasítás a megadott elemek után történik!
        aktkezdet=None
        akthossz=0

print("A Kimotripszin enzimmel széthasított BSA lánc leghosszabb darabja\n")
print(f'A szakasz előtti hasító aminosav a {maxkezdet}. sorszámú {bsa[maxkezdet-1]}.')
print(f'A kezdő aminosav a {maxkezdet+1}. sorszámú {bsa[maxkezdet]}, a szakasz hossza: {maxhossz},')
print(f'az utolsó, a hasító elem a {maxkezdet+maxhossz}. sorszámú {bsa[maxkezdet+maxhossz-1]}.')


print("\n6. feladat\n")

hasitasok=list(map(lambda i : bsa[i]=="R" and (bsa[i+1]=="A" or bsa[i+1]=="V") ,range(len(bsa))))
# minden indexhez hozzárendeljük, hogy történt-e hasítás a bsa-ban az adott helyen

if True in hasitasok:
    elsohasitas=hasitasok.index(True)
    db=bsa[:elsohasitas].count("C")
    print(f'Az első hasítási láncban {db} Cisztein található.')
else:
    print("A Factor XI enzim nem hasította a BSA-láncot")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2006maj/EmeltInfo2006maj.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

2. feladat

A relatív molekulatömegek meghatározása befejeződött.

3. feladat

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 kiíratás és a szövegfájl lezárása sikeresen befejeződött.

4. feladat

A BSA nevű fehérje összegképlete: C 2923 H 4594 O 895 N 778 S 39

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

5. feladat

A Kimotripszin enzimmel széthasított BSA lánc leghosszabb darabja

A szakasz előtti hasító aminosav a 260. sorszámú Y.
A kezdő aminosav a 261. sorszámú I, a szakasz hossza: 46,
az utolsó, a hasító elem a 306. sorszámú F.

6. feladat

Az első hasítási láncban 12 Cisztein található.

A befejezéshez nyomd meg az ENTER billentyűt!
```

```
Az eredmeny.txt szovegfájl

3. feladat
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

4. feladat
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!

```
"""
2006. október: Zenei adók
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("musor.txt")

adok=[]
percek=[]
mpek=[]
azok=[]

z = int(betxt.readline().strip())

for i in range(z):
    sor=betxt.readline().strip()
    s=sor.index(" ")
    adok.append(int(sor[:s]))
    sor=sor[s+1:]
    s=sor.index(" ")
    percek.append(int(sor[:s]))
    sor=sor[s+1:]
    s=sor.index(" ")
    mpek.append(int(sor[:s]))
    azok.append(sor[s+1:])

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")


print("\n2. feladat\n")

for i in range(1,4):
    print(f'A(z) {i}. adón {adok.count(i)} dal hallható.')


print("\n3. feladat\n")

def konvertalo(t):
    mp=t%60
    t=t//60
    perc=t%60
    ora=t//60
    return f'{ora:02}:{perc:02}:{mp:02}'  # bevezető nullák

akt=[0,0,0]

kezdetek=[]
vegek=[]

for i,ado in enumerate(adok):
    kezdetek.append(akt[ado-1])
    akt[ado-1]+=60*percek[i]+mpek[i]
    vegek.append(akt[ado-1])

EC=list(filter(lambda i:("Eric Clapton" in azok[i]) and (adok[i]==1),range(len(adok))))

if len(EC)>=2:
    elsokezd=kezdetek[EC[0]]
    utolsoveg=vegek[EC[-1]]
    ido=utolsoveg-elsokezd
    print(f'Az első Eric Clapton szám kezdete ({konvertalo(elsokezd)})\n\
és az utolsó vége ({konvertalo(utolsoveg)})\n\
között eltelt idő {konvertalo(ido)} az 1. adón.')
else:
    print("Nem játszottak két Eric Clapton számot az 1. adón.")
```

```
print("\n4. feladat\n")

voltado=[False,False,False]
"""
A feladat szerint feltételezhetjük, hogy amikor az "Omega:Legenda" dal kezdődik valamelyik adón,
akkor a többin is van még adás. Ez sajnos nem teljesül, ezért megvizsgálom.
Azt viszont az egyszerűség kedvéért feltételezem, hiszen a listából is jól látszik,
hogy már mindegyik adón volt a megadott dal előtt valamilyen dal.
"""
if "Omega:Legenda" in azok:
    aktindex=azok.index("Omega:Legenda")
    aktado=adok[aktindex]
    voltado[aktado-1]=True
    aktkezdet=kezdetek[aktindex]
    print(f'Az) "Omega:Legenda" dal a(z) {aktado}. adón kezdődött {konvertalo(aktkezdet)} időpontban.')
    egyik=list(filter(lambda i:(kezdetek[i]<aktkezdet) and (adok[i]!=aktado),range(len(adok))))
    egyikdal=azok[egyik[-1]]
    egyikado=adok[egyik[-1]]
    voltado[egyikado-1]=True
    print(f'A(z) {egyikado}. adón a(z) "{egyikdal}" dal kezdődött előtte utoljára,')
    if vegek[egyik[-1]]>aktkezdet:
        print(f'{konvertalo(kezdetek[egyik[-1]])} időpontban kezdődött és még hallható is volt.')
    else:
        print(f'de már {konvertalo(vegek[egyik[-1]])} időpontban véget ért, az adó műsora befejeződött.')
    masikado=voltado.index(False)+1
    masik=list(filter(lambda i:(kezdetek[i]<aktkezdet) and (adok[i]==masikado),range(len(adok))))
    masikdal=azok[masik[-1]]
    print(f'A(z) {masikado}. adón a(z) "{masikdal}" dal kezdődött előtte utoljára,')
    if vegek[masik[-1]]>aktkezdet:
        print(f'{konvertalo(kezdetek[masik[-1]])} időpontban kezdődött és még hallható is volt.')
    else:
        print(f'de már {konvertalo(vegek[masik[-1]])} időpontban véget ért, az adó műsora befejeződött.')
else:
    print('Nem játszották egyik adón sem az "Omega:Legenda" számot')


print("\n5. feladat\n")

def talal(mit,miben):
    mit=mit.lower()
    miben=miben.lower()
    van=True
    i=0
    while van and i<len(mit):
        if mit[i] in miben:
            aktindex=miben.index(mit[i])
            miben=miben[aktindex+1:]
        else:
            van=False
        i+=1
    return van

kitxt = open("keres.txt","w")

foszlany=input("Kérem a felismerhető karaktereket: ")
foszlany=foszlany.strip().lower()

kitxt.write(f'{foszlany}\n')

for az in azok:
    if talal(foszlany,az):
        kitxt.write(f'{az}\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.")
```

```
print("\n6. feladat\n")

kezd=0
veg=180 # 0 órakor hírekkel kezd
ora=0

for i,ado in enumerate(adok):
    if ado==1:
        kezd=veg
        hossz=60+percek[i]*60+mpek[i]
        veg=kezd+hossz
        if veg>(ora+1)*3600:
            ora+=1
            kezd=ora*3600+180
            veg=kezd+hossz

print(f'Az új műsorszerkezetben az 1. adó műsorának befejezési időpontja: \
{konvertalo(veg)}')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

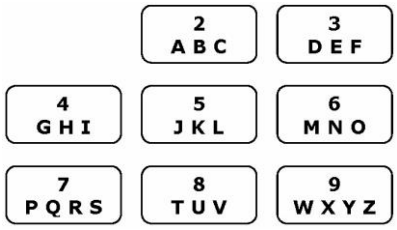
```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2006okt/EmeltInfo2006okt.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

2. feladat

A(z) 1. adón 232 dal hallható.
A(z) 2. adón 215 dal hallható.
A(z) 3. adón 230 dal hallható.

3. feladat

Az első Eric Clapton szám kezdete (00:05:03)
és az utolsó vége (17:56:15)
között eltelt idő 17:51:12 az 1. adón.

4. feladat

Az) "Omega:Legenda" dal a(z) 3. adón kezdődött 16:45:08 időpontban.
A(z) 1. adón a(z) "Eric Clapton:Grand Illusion" dal kezdődött előtte utoljára,
16:41:30 időpontban kezdődött és még hallható is volt.
A(z) 2. adón a(z) "Eric Clapton:Pretty Girl" dal kezdődött előtte utoljára,
de már 15:31:28 időpontban véget ért, az adó műsora befejeződött.

5. feladat

Kérem a felismerhető karaktereket: gAoaf

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

6. feladat

Az új műsorszerkezetben az 1. adó műsorának befejezési időpontja: 23:44:41

A befejezéshez nyomd meg az ENTER billentyűt!
```

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.

```
"""
2007. május: SMS szavak
@author Klemand66
"""


print("\n1. feladat\n")

def betukodolo(betu):
    betu=betu.lower()
    abc="abcdefghijklmnopqrstuvwxyz"
    kod="22233344455566677778889999"
    aktindex=abc.index(betu)
    return kod[aktindex]

betu=input("Kérek egy betűt: ")

print(f'A megadott {betu} betű kódja {betukodolo(betu)}')


print("\n2. feladat\n")

def szokodolo(szo):
    kod=""
    for betu in szo:
        kod+=betukodolo(betu)
    return kod

szo=input("Kérek egy szót: ")

print(f'A megadott {szo} szó kódja {szokodolo(szo)}')


print("\n3. feladat\n")

betxt = open("szavak.txt")
szavak=[]

for sor in betxt:
    szavak.append(sor.strip())

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")


print("\n4. feladat\n")

hosszak=list(map(lambda szo: len(szo),szavak))
maxhossz=max(hosszak)
leghosszabbak=list(filter(lambda szo: len(szo)==maxhossz,szavak))

print(f'A leghosszabb szó (szavak) hossza {maxhossz}')
print(f'A leghosszabb szó (szavak): ',end="")

for szo in leghosszabbak:
    print(f'{szo} ',end="")
print()


print("\n5. feladat\n")

rovidek=list(filter(lambda szo: len(szo)<6,szavak))

print(f'A fájlban {len(rovidek)} rövid szó található.')


print("\n6. feladat\n")

kitxt = open("kodok.txt","w")

for szo in szavak:
    kitxt.write(f'{szokodolo(szo)}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")
```

```
print("\n7. feladat\n")

szamsor=input("Kérek egy számsort: ")
aktkod=szamsor.strip()

kodok=[]

for szo in szavak:
    kodok.append(szokodolo(szo))

jok=list(filter(lambda i: kodok[i]==aktkod,range(len(szavak))))

print(f'\nA megadott {aktkod} számsorhoz tartozó szavak: ')

for i in jok:
    print(f'{szavak[i]} ',end="")
print()


print("\n8. feladat\n")

kodlista=[]
szolista=[]

for i,kod in enumerate(kodok):
    if kod in kodlista:
        aktindex=kodlista.index(kod)
        szolista[aktindex].append(szavak[i])
    else:
        kodlista.append(kod)
        szolista.append([szavak[i]])

sor=""

for i,kod in enumerate(kodlista):
    if len(szolista[i])>1:
        for szo in szolista[i]:
            sor+=f'{szo} : {kod}; '
            if len(sor)>80:
                print(sor)
                sor=""
print(sor)


print("\n9. feladat\n")

szodb=list(map(lambda i: len(szolista[i]),range(len(szolista))))
maxhossz=max(szodb)
leggazdagabbak=list(filter(lambda i: len(szolista[i])==maxhossz,range(len(szolista))))

print(f'A leggazdagabb kód(ok) szavainak száma {maxhossz}')
print(f'A kód(ok) és szavai(k): ')

for i in leggazdagabbak:
    print(f'{kodlista[i]} ',end="")
    for szo in szolista[i]:
        print(f'{szo} ',end="")
    print()
print()


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2007maj/EmeltInfo2007maj.py ==================

1. feladat

Kérek egy betűt: G
A megadott G betű kódja 4

2. feladat

Kérek egy szót: kalaP
A megadott kalaP szó kódja 52527

3. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

4. feladat

A leghosszabb szó (szavak) hossza 13
A leghosszabb szó (szavak): megfeledkezik

5. feladat

A fájlban 279 rövid szó található.

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

7. feladat

Kérek egy számsort: 5662

A megadott 5662 számsorhoz tartozó szavak:
koma lomb

8. feladat

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;

9. feladat

A leggazdagabb kód(ok) szavainak száma 3
A kód(ok) és szavai(k):
42727 garas harap hasas
527 kap kar lap


A befejezéshez nyomd meg az ENTER billentyűt!
```

A kodok.txt szöveges fájl

```
22525
2276627
23
244
244632566
2525
…
```

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 ...
   ```

```
"""
2007. október: Foci
@author Klemand66
"""

print("\n1. feladat\n")

ford=[]
hg=[]
vg=[]
hgf=[]
vgf=[]
hcs=[]
vcs=[]

betxt = open("meccs.txt")

n = int(betxt.readline().strip())

for i in range(n):
    darsor=betxt.readline().strip().split()
    ford.append(int(darsor[0]))
    hg.append(int(darsor[1]))
    vg.append(int(darsor[2]))
    hgf.append(int(darsor[3]))
    vgf.append(int(darsor[4]))
    hcs.append(darsor[5])
    vcs.append(darsor[6])

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")


print("\n2. feladat\n")

aktford=int(input("Kérem egy forduló sorszámát: "))
print()

for i,f in enumerate(ford):
    if f==aktford:
        print(f' {hcs[i]}-{vcs[i]}: {hg[i]}-{vg[i]} ({hgf[i]}-{vgf[i]})')


print("\n3. feladat\n")

for i,f in enumerate(ford):
    if (hg[i]-vg[i])*(hgf[i]-vgf[i])<0:
        if hg[i]>vg[i]:
            print(f' A(z) {f:2}. fordulóban fordított és nyert a hazai  {hcs[i]}.')
        else:
            print(f' A(z) {f:2}. fordulóban fordított és nyert a vendég {vcs[i]}.')


print("\n4. feladat\n")

aktcsapat=input("Kérem egy csapat nevét: ").strip().capitalize()


print("\n5. feladat\n")

lott=0
kapott=0

for i in range(len(ford)):
    if hcs[i]==aktcsapat:
        lott+=hg[i]
        kapott+=vg[i]
    elif vcs[i]==aktcsapat:
        lott+=vg[i]
        kapott+=hg[i]

print(f' A(z) {aktcsapat} összesen {lott} gólt lőtt és {kapott} gólt kapott.')
```

```
print("\n6. feladat\n")

kikaptak=False
for i,f in enumerate(ford):
    if hcs[i]==aktcsapat and vg[i]>hg[i]:
        kikaptak=True
        print(f' A(z) {aktcsapat} csapat a(z) {f}. fordulóban kapott ki először otthon \
a(z) {vcs[i]} csapattól.')
        break

if not kikaptak:
    print(f' A(z) {aktcsapat} csapat otthon veretlen maradt.')


print("\n7. feladat\n")

eredmenylista=[]
gyaklista=[]

for i in range(len(ford)):
    a=max(hg[i],vg[i])
    b=min(hg[i],vg[i])
    eredmeny=f'{a}-{b}'
    if eredmeny in eredmenylista:
        aktindex=eredmenylista.index(eredmeny)
        gyaklista[aktindex]+=1
    else:
        eredmenylista.append(eredmeny)
        gyaklista.append(1)

kitxt = open("stat.txt","w")

for i,eredmeny in enumerate(eredmenylista):
    kitxt.write(f'{eredmeny}: {gyaklista[i]} darab\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2007okt\EmeltInfo2007okt.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

2. feladat

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)

3. feladat

 A(z)  8. fordulóban fordított és nyert a vendég Jelmezesek.
 A(z)  7. fordulóban fordított és nyert a hazai  Bogarak.
 A(z) 11. fordulóban fordított és nyert a vendég Ijedtek.
 A(z) 14. fordulóban fordított és nyert a vendég Kedvesek.
 A(z)  7. fordulóban fordított és nyert a vendég Lelkesek.

4. feladat

Kérem egy csapat nevét: lelkesek

5. feladat

 A(z) Lelkesek összesen 13 gólt lőtt és 21 gólt kapott.

6. feladat

 A(z) Lelkesek csapat a(z) 6. fordulóban kapott ki először otthon a(z) Csikosak csapattól.

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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!

```
"""
2007. május: SMS
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("sms.txt")

orak=[]
percek=[]
telszamok=[]
uzenetek=[]

k = int(betxt.readline().strip())

for i in range(k):
    sor=betxt.readline().strip()
    darsor=sor.split()
    orak.append(int(darsor[0]))
    percek.append(int(darsor[1]))
    telszamok.append(darsor[2])
    sor=betxt.readline().strip()
    uzenetek.append(sor)

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")


print("\n2. feladat\n")

db=len(uzenetek)
if db<=10:
    print(uzenetek[db-1]) # az érkező utolsó üzenet lesz a legfrissebb
else:
    print(uzenetek[9]) # a 10. üzenet lesz a legfrissebb


print("\n3. feladat\n")

hosszak=list(map(lambda uzenet: len(uzenet), uzenetek))
maxhossz=max(hosszak)
print(f'A leghosszabb üzenet hossza {maxhossz}.')
maxindex=hosszak.index(maxhossz)
print(f'{orak[maxindex]}, {percek[maxindex]}, {telszamok[maxindex]}, {uzenetek[maxindex]}\n')

minhossz=min(hosszak)
print(f'A legrovidebb üzenet hossza {minhossz}.')
minindex=hosszak.index(minhossz)
print(f'{orak[minindex]}, {percek[minindex]}, {telszamok[minindex]}, {uzenetek[minindex]}')

"""
A feladat szerint elég volt egyet-egyet is megadni, de könnyen megadhatnánk az összeset.
(Ezért is tetszik jobban ez a módszer, mint a sima maximumkiválasztás programozási tétel.)

maxindexek=list(filter(lambda i: hosszak[i]==maxhossz,range(db)))
for i in maxindexek:
    print(f'{orak[i]}, {percek[i]}, {telszamok[i]}, {uzenetek[i]}')
print()

minindexek=list(filter(lambda i: hosszak[i]==minhossz,range(db)))
for i in minindexek:
    print(f'{orak[i]}, {percek[i]}, {telszamok[i]}, {uzenetek[i]}')
print()
"""

print("\n4. feladat\n")

print("Karakterhossz szerinti statisztika:")

for k in range(5):
    stat=list(filter(lambda i: hosszak[i]>k*20 and hosszak[i]<=(k+1)*20 ,range(db)))
    print(f'{k*20+1:2} - {(k+1)*20:3}: {len(stat):2}')
```

```
print("\n5. feladat\n")

dbsz=0
for i in range(24):
    aktdb=orak.count(i)
    if aktdb>10:
        dbsz+=aktdb-10 # minden órában a 10 fölöttiek maradnak bent
print(f'Ernőnek összesen {dbsz} db üzenetet kell a szolgáltatótól kikérnie.')


print("\n6. feladat\n")

# Kiválogatjuk a barátnő smseit:
bnosmsek=list(filter(lambda i:telszamok[i]=="123456789" ,range(db)))

if len(bnosmsek)<2:
    print("Nincs elegendő üzenet.")
else:
    # Meghatározzuk a barátnő küldésidőit:
    bnoidok=list(map(lambda i:orak[i]*60+percek[i] ,bnosmsek))
    # Meghatározzuk a küldések közötti időket:
    koztesidok=list(map(lambda i:bnoidok[i]-bnoidok[i-1] ,range(1,len(bnoidok))))
    maxido=max(koztesidok)
    print(f'A leghosszabb idő barátnői smsek nélkül {maxido//60} óra {maxido%60} perc volt.')


print("\n7. feladat\n")

sor=input("Kérem az órát, a percet és a telefonszámot szóközökkel elválasztva: ")

darsor=sor.strip().split()
orak.append(int(darsor[0]))
percek.append(int(darsor[1]))
telszamok.append(darsor[2])

sor=input("Kérem az üzenetet: ")

uzenetek.append(sor.strip())


print("\n8. feladat\n")

"""
Először elkészítjük a telefonszámok listáját
a hozzájuk tartozó smsekkel a kiírásnak megfelelő formában,
majd a rendezéshez a listákat egy kétdimenziós listává egyesítjük.
"""

telszamlista=[] # sima lista
smslista=[]     # sms-listák listája

for i,telszam in enumerate(telszamok):
    sms=str(orak[i])+" "+str(percek[i])+" "+uzenetek[i]
    if telszam in telszamlista:
        aktindex=telszamlista.index(telszam)
        smslista[aktindex].append(sms) # itt csak hozzáfűzzük az smst az smsek megfelelő listájához
    else:
        telszamlista.append(telszam)
        smslista.append([sms]) # itt nyitjuk az smsek új listáját

egylista=[]
for i in range(len(telszamlista)):
    egylista.append([telszamlista[i],smslista[i]])

egylista.sort(key=lambda elem: elem[0])

kitxt = open("smski.txt","w")

for elem in egylista:
    kitxt.write(f'{elem[0]}\n')
    for sms in elem[1]:
        kitxt.write(f' {sms}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2008maj/EmeltInfo2008maj.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

2. feladat

Erdekli Ont egy telefon? A vezetek nelkuli telefon most a legolcsobb!

3. feladat

A leghosszabb üzenet hossza 83.
9, 40, 434325432, A nyelvvizsgadra mennyi potlekot kapsz? Nekem meg nem fizettek egy fillert sem. :-(

A legrovidebb üzenet hossza 13.
11, 36, 434324223, Kesz a kocsi!

4. feladat

Karakterhossz szerinti statisztika:
 1 -  20:  5
21 -  40: 17
41 -  60:  2
61 -  80:  5
81 - 100:  1

5. feladat

Ernőnek összesen 3 db üzenetet kell a szolgáltatótól kikérnie.

6. feladat

A leghosszabb idő barátnői smsek nélkül 1 óra 27 perc volt.

7. feladat

Kérem az órát, a percet és a telefonszámot szóközökkel elválasztva: 20 0 123456789
Kérem az üzenetet: Mikor irsz mar, dragam?

8. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

Az smski.txt szoveges fájl

```
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!
 20 0 Mikor irsz mar, dragam?
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!

```
"""
2008. október: Robot
@author Klemand66
"""

from math import *

print("\n1. feladat\n")

betxt = open("program.txt")
n = int(betxt.readline().strip())

programok=[]

for i in range(n):
    sor=betxt.readline().strip()
    programok.append(sor)

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.\n")


print("\n2. feladat\n")

k=int(input(f'Kérem egy program sorszámát (1 - {n}-ig): '))

aktpr = programok[k-1]

print("\na. feladat\nA program ",end="")

if "ED" in aktpr or "DE" in aktpr or "KN" in aktpr or "NK" in aktpr:
    print("egyszerűsíthető.")
else:
    print("nem egyszerűsíthető.")

print("\nb. feladat")
EDdb=abs(aktpr.count("E")-aktpr.count("D"))
KNdb=abs(aktpr.count("K")-aktpr.count("N"))

print(f'{EDdb} lépést kell tenni az ED, {KNdb} lépést a KN tengely mentén.')

print("\nc. feladat\n")

def tav(pr): # túl hosszú lett volna egy kifejezésként a lambda függvénybe írni
    x=pr.count("E")-pr.count("D")
    y=pr.count("K")-pr.count("N")
    return sqrt(x**2+y**2) # ehhez kellett a math modul

tavok=list(map(lambda i: tav(aktpr[:i+1]),range(len(aktpr))))

maxtav=max(tavok)
lepes=tavok.index(maxtav)+1

print(f'A(z) {lepes}. lépés után volt legtávolabb: {maxtav:.3f} cm távolságra.\n')


print("\n3. feladat\n")

print("A kis akkut igénylő programok és energiaszükségletük:")

for i,pr in enumerate(programok):
    energia=2+len(pr)
    for j in range(1,len(pr)):
        if pr[j]!=pr[j-1]:
            energia+=2
    if energia<=100:
        print(f'{i+1} {energia}')
print()
```

```
print("\n4. feladat\n")

kitxt = open("ujprog.txt","w")

for pr in programok:
    pr+="X" # így a program végét is tudja kezelni
    ujpr=""
    irany=pr[0]
    db=1
    for j in range(1,len(pr)):
        if pr[j]==irany:
            db+=1
        else:
            if db==1:
                ujpr+=irany
            else:
                ujpr+=str(db)+irany
                db=1
            irany=pr[j]
    kitxt.write(f'{ujpr}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")


print("\n5. feladat\n")

ujpr=input("Kérek egy új típusú programot (max. 200 ismétlődéssel): ")

rpr=""
db=0

for i in range(len(ujpr)):
    if ujpr[i].isnumeric():
        db=10*db+int(ujpr[i])
    else:
        if db>0:
            rpr+=db*ujpr[i]
            db=0
        else:
            rpr+=ujpr[i]

print(f'A program régi alakja: {rpr}')

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2008okt/EmeltInfo2008okt.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.


2. feladat

Kérem egy program sorszámát (1 - 13-ig): 10

a. feladat
A program egyszerűsíthető.

b. feladat
8 lépést kell tenni az ED, 11 lépést a KN tengely mentén.

c. feladat

A(z) 159. lépés után volt legtávolabb: 13.601 cm távolságra.


3. feladat

A kis akkut igénylő programok és energiaszükségletük:
1 29
2 52
3 84


4. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.


5. feladat

Kérek egy új típusú programot (max. 200 ismétlődéssel): E2N4DENDEND2E2D2NKED
A program régi alakja: ENNDDDDENDENDEEDDNNKED

A befejezéshez nyomd meg az ENTER billentyűt!
```

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
…
```

```
"""
2009. május: Lift
@author Klemand66
"""

from random import *


print("\n1. feladat\n")

betxt = open("igeny.txt")

szdb = int(betxt.readline().strip()) # szintek száma
csdb = int(betxt.readline().strip()) # csapatok száma
igdb = int(betxt.readline().strip()) #igények száma

# Mivel rendezésre nem lesz szükség, külön listákat alkalmazunk.
orak, percek, mpek, csapatok, iszintek, cszintek=[],[],[],[],[],[]

for i in range(igdb):
    sor=betxt.readline().strip()  # beolvasás és megtisztítás
    darsor=sor.split()            # darabolás
    adatsor=list(map(int,darsor)) # egész számmá alakítás
    orak.append(adatsor[0])
    percek.append(adatsor[1])
    mpek.append(adatsor[2])
    csapatok.append(adatsor[3])
    iszintek.append(adatsor[4])
    cszintek.append(adatsor[5])

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

szint0=int(input(f'Kérem a 9 órai kezdőszintet (max {szdb}): '))



print("\n3. feladat\n")

print(f'A lift a(z) {cszintek[-1]}. szinten áll az utolsó igény teljesítése után.')



print("\n4. feladat\n")

minsz=min(szint0,min(iszintek),min(cszintek))
maxsz=max(szint0,max(iszintek),max(cszintek))

print(f'A legalacsonyabb szint: {minsz}')
print(f'A legmagasabb szint: {maxsz}')



print("\n5. feladat\n")

aktszint=szint0
uresdb=0
utasdb=0

for i in range(igdb):
    if iszintek[i]>aktszint:
        uresdb+=1
    if cszintek[i]>iszintek[i]:
        utasdb+=1
    aktszint=cszintek[i]

print(f'Utas nélkül {uresdb}, utassal {utasdb} alkalommal kellett a liftnek felfelé menni.')
```

```
print("\n6. feladat\n")

print("A következő csapatok nem vették igénybe a liftet: ", end="")
for i in range(1, csdb+1):
    if not i in csapatok:
        print(f'{i} ',end="")
print()


print("\n7. feladat\n")

aktcsapat=randrange(1,csdb+1)

aktcsapatindexek=list(filter(lambda i:csapatok[i]==aktcsapat, range(igdb)))

biz=False

for i in range(1,len(aktcsapatindexek)):
    if iszintek[aktcsapatindexek[i]]!=cszintek[aktcsapatindexek[i-1]]:
        print(f'A(z) {aktcsapat}. csapat esetében bizonyítható a szabálytalanság.')
        print(f'Gyalog közlekedtek a(z) {cszintek[aktcsapatindexek[i-1]]}. \
és {iszintek[aktcsapatindexek[i]]}. emeletek között.')
        biz=True
        break
if not biz:
    print(f'A(z) {aktcsapat}. csapat esetében nem bizonyítható a szabálytalanság.')



print("\n8. feladat\n")

kitxt = open("blokkol.txt","w")

for i in range(igdb):
    if csapatok[i]== aktcsapat:
        kitxt.write(f'Befejezés ideje: {orak[i]}:{percek[i]}:{mpek[i]}\n')
        sik=input("Kérem a munka eredményét (befejezett vagy befejezetlen): ")
        kitxt.write(f'Sikeresség: {sik}\n')
        kitxt.write(f'----\n')
        kitxt.write(f'Indulási emelet: {iszintek[i]}\n')
        kitxt.write(f'Célemelet: {cszintek[i]}\n')
        fkod=input("Kérem a következő munka feladatkódját (1 és 99 közötti szám): ")
        kitxt.write(f'Feladatkód: {fkod}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2009maj\EmeltInfo2009maj.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

2. feladat

Kérem a 9 órai kezdőszintet (max 65): 4

3. feladat

A lift a(z) 10. szinten áll az utolsó igény teljesítése után.

4. feladat

A legalacsonyabb szint: 4
A legmagasabb szint: 55

5. feladat

Utas nélkül 37, utassal 57 alkalommal kellett a liftnek felfelé menni.

6. feladat

A következő csapatok nem vették igénybe a liftet: 6 9 17 23 25

7. feladat

A(z) 12. csapat esetében bizonyítható a szabálytalanság.
Gyalog közlekedtek a(z) 26. és 27. emeletek között.

8. feladat

Kérem a munka eredményét (befejezett vagy befejezetlen): befejezett
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 22
Kérem a munka eredményét (befejezett vagy befejezetlen): befejezetlen
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 88
Kérem a munka eredményét (befejezett vagy befejezetlen): befejezett
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 36
Kérem a munka eredményét (befejezett vagy befejezetlen): befejezetlen
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 56
Kérem a munka eredményét (befejezett vagy befejezetlen): befejezetlen
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 66
Kérem a munka eredményét (befejezett vagy befejezetlen): befejezett
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 48
Kérem a munka eredményét (befejezett vagy befejezetlen): befejezetlen
Kérem a következő munka feladatkódját (1 és 99 közötti szám): 6
A kiíratás és a szövegfájl lezárása sikeresen befejeződött.


A befejezéshez nyomd meg az ENTER billentyűt!
```

```
A blokkol.txt szövegfájl


Befejezés ideje: 9:8:19
Sikeresség: befejezett
----
Indulási emelet: 5
Célemelet: 9
Feladatkód: 22
Befejezés ideje: 9:38:20
Sikeresség: befejezetlen
----
Indulási emelet: 9
Célemelet: 23
Feladatkód: 88
Befejezés ideje: 10:22:24
Sikeresség: befejezett
----
Indulási emelet: 23
Célemelet: 26
Feladatkód: 36
Befejezés ideje: 11:9:41
Sikeresség: befejezetlen
----
Indulási emelet: 27
Célemelet: 33
Feladatkód: 56
Befejezés ideje: 11:59:58
Sikeresség: befejezetlen
----
Indulási emelet: 33
Célemelet: 22
Feladatkód: 66
Befejezés ideje: 12:56:40
Sikeresség: befejezett
----
Indulási emelet: 22
Célemelet: 11
Feladatkód: 48
Befejezés ideje: 13:59:48
Sikeresség: befejezetlen
----
Indulási emelet: 5
Célemelet: 10
Feladatkód: 6
```

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 „`kevés 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.

```
"""
2009. május idegennyelvű: Automata
@author Klemand66
"""


print("\n1. feladat\n")

betxt = open("csoki.txt")

rekdb = int(betxt.readline().strip())
csrekeszek, csdbk, csarak = [], [], []
for i in range(rekdb):
    sor=betxt.readline().strip()  # beolvasás és megtisztítás
    darsor=sor.split()            # darabolás
    adatsor=list(map(int,darsor)) # egész számmá alakítás
    csrekeszek.append(adatsor[0])
    csdbk.append(adatsor[1])
    csarak.append(adatsor[2])

betxt.close()


betxt = open("vasarlas.txt")

vasdb = int(betxt.readline().strip())
vrekeszek, vdbk, vcimletdbk = [], [], []
for i in range(vasdb):
    sor=betxt.readline().strip()
    darsor=sor.split()
    adatsor=list(map(int,darsor))
    vrekeszek.append(adatsor[0])
    vdbk.append(adatsor[1])
    vcimletdbk.append(adatsor[2:])

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")


print("\n2. feladat\n")

ertek=0

for i in range(rekdb):
    ertek+=csdbk[i]*csarak[i]

print(f'Az automatában {ertek} fabatka értékű csokoládé van.')


print("\n3. feladat\n")

prekeszek=[]

for rek in vrekeszek:
    if not rek in prekeszek:
        prekeszek.append(rek)

prekeszek.sort() # ha sztringként rendezném, a 18 a 2 előtt lenne
strprekeszek=list(map(str,prekeszek))

print(f'A következő rekeszekből próbáltak vásárolni: {" ".join(strprekeszek)}')


print("\n4. feladat\n")

keret=int(input("Kérem a 7 egyforma csokoládéra szánt összeget: "))

print("A következő rekeszekből lehet választani: ", end="")
for i in range(rekdb):
    if csdbk[i] >= 7 and 7*csarak[i]<=keret:
        print(csrekeszek[i],end=" ")
print()
```

```
print("\n5. feladat\n")

r=int(input(f'Kérem a rekesz sorszámát (max: {rekdb}): '))
db=int(input("Kérem a csokoládék darabszámát: "))
aktindex=csrekeszek.index(r)
osszeg=csarak[aktindex]*db

print(f'A fizetendő összeg: {osszeg} fabatka.')
print("Az optimális címletezés:")
cimletek=[1, 2, 5, 10, 20, 50, 100 ]
cimletdbk = [ 0, 0, 0, 0, 0, 0, 0 ]

while osszeg>0:
    for i in range(6,-1,-1):
        if cimletek[i]<=osszeg:
            cimletdbk[i]+=1
            osszeg-=cimletek[i]
            break

for i in range(6,-1,-1):
    if cimletdbk[i]!=0:
        print(f'{cimletek[i]:3}: {cimletdbk[i]:2}')


print("\n6. feladat\n")

# Általánosítva tetszőleges rekeszre

r=int(input(f'Kérem egy rekesz sorszámát (max: {rekdb}, a feladat szerint 7): '))

kitxt = open(f'rekesz{r}.txt',"w")

ind=csrekeszek.index(r)
csdb=csdbk[ind]
csar=csarak[ind]

for i in range(vasdb):
    if vrekeszek[i]==r:
        kitxt.write(f'{i+1:2}')
        aktdb=vdbk[i]
        aktcimletdbk=vcimletdbk[i]
        bepenz=sum(list(map(lambda j:aktcimletdbk[j]*cimletek[j],range(7))))
        if csdb>=aktdb and bepenz>=aktdb*csar:
            kitxt.write(f'\t{aktdb} (visszajár: {bepenz}-{aktdb}*{csar}={bepenz-aktdb*csar})')
            csdb-=aktdb
        else:
            if csdb<aktdb:
                kitxt.write(f'\tkevés a csoki ({csdb}<{aktdb})')
            if bepenz<aktdb*csar:
                kitxt.write(f'\tnem volt elég pénz ({bepenz}<{aktdb}*{csar})')
        kitxt.write("\n")

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2009mid\EmeltInfo2009mid.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

2. feladat

Az automatában 24817 fabatka értékű csokoládé van.

3. feladat

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

4. feladat

Kérem a 7 egyforma csokoládéra szánt összeget: 500
A következő rekeszekből lehet választani: 17 20 24

5. feladat

Kérem a rekesz sorszámát (max: 25): 2
Kérem a csokoládék darabszámát: 8
A fizetendő összeg: 984 fabatka.
Az optimális címletezés:
100:  9
 50:  1
 20:  1
 10:  1
  2:  2

6. feladat

Kérem egy rekesz sorszámát (max: 25, a feladat szerint 7): 7
A kiíratás és a szövegfájl lezárása sikeresen befejeződött.


A befejezéshez nyomd meg az ENTER billentyűt!
```

A rekesz7.txt szövegfájl

```
 2      1 (visszajár: 110-1*93=17)
 9      1 (visszajár: 100-1*93=7)
21      kevés a csoki (1<4)     nem volt elég pénz (370<4*93)
46      1 (visszajár: 100-1*93=7)
47      kevés a csoki (0<4)     nem volt elég pénz (370<4*93)
55      kevés a csoki (0<6)     nem volt elég pénz (550<6*93)
```

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!

```
"""
2009. október: Útépítés
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("forgalom.txt")
jdb = int(betxt.readline().strip())

jarmuvek=[]
for i in range(jdb):
    sor=betxt.readline().strip()
    darsor=sor.split()
    jarmu=[int(darsor[0]),int(darsor[1]),int(darsor[2])]
    # jarmu[    0: óra,      1: perc,       2: mp,
    erkezes=3600*jarmu[0]+60*jarmu[1]+jarmu[2]
    jarmu.append(erkezes)  # 3: érkezés,
    jarmu+=[int(darsor[3]), darsor[4]]
       # 4: áthaladási idő,  5: honnan]
    jarmuvek.append(jarmu)

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")


print("\n2. feladat\n")

n=int(input(f'Kérem a jármű sorszámát (n<={jdb}): '))

if jarmuvek[n-1][5]=="A":
    irany="Felső"
else:
    irany="Alsó"

print(f'A(z) {n}. belépő jármű {irany} felé haladt.')


print("\n3. feladat\n")

felsobe=list(filter(lambda i:jarmuvek[i][5]=="A" , range(jdb)))

if len(felsobe)>1:
    u=felsobe[-1]
    ue=felsobe[-2]
    idokul=jarmuvek[u][3]-jarmuvek[ue][3]

    print(f'A Felső felé haladó utolsó két jármű érkezése között {idokul} mp különbség volt.')
else:
    print("Nem haladt két jármű Felső város irányába.")


print("\n4. feladat\n")

dba,dbf=[],[]
for i in range(24):
    dba.append(0)
    dbf.append(0)

for jarmu in jarmuvek:
    if jarmu[5]=="A":
        dba[jarmu[0]]+=1
    else:
        dbf[jarmu[0]]+=1

print("Az óránként Alsó, ill. Felső város felől érkező autók száma: ")

for i in range(24):
    if dba[i]+dbf[i]!=0:
        print(f'{i:2} {dba[i]:3} {dbf[i]:3} ')
```

```
print("\n5. feladat\n")

rjarmuvek=sorted(jarmuvek,key=lambda jarmu: jarmu[4])

print("A 10 leggyorsabb jármű: ")

for i,jarmu in enumerate(rjarmuvek):
    if jarmu[5]=="A":
        honnan="Alsó"
    else:
        honnan="Felső"
    print(f'{jarmu[0]:2} {jarmu[1]:2} {jarmu[2]:2} {honnan:5} {1000/jarmu[4]:.1f} m/s')
    if i==9:
        break


print("\n6. feladat\n")

def konvertalo(t):
    mp=t%60
    t=t//60
    perc=t%60
    ora=t//60
    return f'{ora} {perc} {mp}'

felsobe=list(filter(lambda i:jarmuvek[i][5]=="F" , range(jdb)))

kitxt = open("also.txt","w")

kilepes=-1 # irreálisan kicsi kezdőérték

for ind in felsobe:
    idki=jarmuvek[ind][3] + jarmuvek[ind][4] # forgalom nélküli ideális kilépés
    if idki > kilepes:
        kilepes=idki
    kitxt.write(f'{konvertalo(kilepes)}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2009okt/EmeltInfo2009okt.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

2. feladat

Kérem a jármű sorszámát (n<=1105): 100
A(z) 100. belépő jármű Alsó felé haladt.

3. feladat

A Felső felé haladó utolsó két jármű érkezése között 228 mp különbség volt.

4. feladat

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

5. feladat

A 10 leggyorsabb jármű:
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

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!


Az also.txt szövegfájl

7 22 1                  7 37 25                 7 50 12
7 23 7                  7 39 25                 7 53 16
7 24 1                  7 42 20                 7 53 31
7 26 15                 7 42 20                 7 54 25
7 26 20                 7 42 20                 7 55 51
7 27 7                  7 43 42                 7 57 30
7 28 9                  7 43 52                 7 58 54
7 28 20                 7 44 34                 7 58 54
7 28 40                 7 45 36                 7 58 54
7 29 46                 7 45 36                 7 59 36
7 29 46                 7 47 4                  7 59 55
7 31 21                 7 47 43                 ...
7 32 37                 7 48 48
7 35 52                 7 48 48
7 36 39                 7 49 34
```

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
…
```

```
"""
2010. május: Helyjegy
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("eladott.txt")

darsor = betxt.readline().strip().split()
adatsor=list(map(int,darsor))
jdb,hossz,ear=adatsor[0],adatsor[1],adatsor[2]
# eladott jegyek száma, úthossz, egységár

helyek,felszok,leszok=[],[],[]
# ülőhelyek, felszállások (km), leszállások (km)

for i in range(jdb):
    darsor=betxt.readline().strip().split()
    adatsor=list(map(int,darsor))
    helyek.append(adatsor[0])
    felszok.append(adatsor[1])
    leszok.append(adatsor[2])

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")


print("\n2. feladat\n")

print(f'Az utolsó jegyvásárló ülésének száma: \
{helyek[-1]}, a beutazott távolság: {leszok[-1]-felszok[-1]} km.')


print("\n3. feladat\n")

print(f'A teljes utat végigutazók listája: ',end="")

vegig=list(filter(lambda i: felszok[i]==0 and leszok[i]==hossz,range(jdb)))
# Kiszűrjük a végigutazók indexeit.
vegigstrlista=list(map(lambda i:str(i+1),vegig))
# A sorszámokat (index+1) sztringgé alakítjuk.
vegigstr=" ".join(vegigstrlista)
# A sztringlista elemeit szóközzel "ragasztva" egy sztringgé fűzzük össze.

print(vegigstr)


print("\n4. feladat\n")

def jegyar(fel,le,ear):
    tav=(le-fel)//10
    if (le-fel)%10>0:
        tav+=1
    ar=tav*ear
    m=ar%5
    ar=ar-m
    if m>2:
        ar+=5
    return ar

jegyarak=list(map(lambda i: jegyar(felszok[i], leszok[i],ear),range(jdb)))

print(f'A társaság bevétele a jegyárakból: {sum(jegyarak)} Ft.')
```

```
print("\n5. feladat\n")

kmekfel, kmekle=[], []

for km in range(hossz+1):
    kmekfel.append(0)
    kmekle.append(0)

for i in range(jdb):
    kmekfel[felszok[i]]+=1
    kmekle[leszok[i]]+=1

megallok=list(filter(lambda i: kmekfel[i]+kmekle[i]>0,range(hossz+1)))

print(f'A végállomás előtti utolsó megálló: {megallok[-2]} km-nél volt.')
print(f'A megállóban {kmekfel[megallok[-2]]} felszálló és {kmekle[megallok[-2]]} leszálló volt.')


print("\n6. feladat\n")

print(f'A busz a kiinduló állomás és a célállomás között {len(megallok)-2} helyen állt meg.')


print("\n7. feladat\n")

aktkm=int(input(f'Kérem az út egy tetszőleges km-ét (km<={hossz}): '))

utasok=list(filter(lambda i: felszok[i]<=aktkm and leszok[i]>aktkm,range(jdb)))
# Az i index az i+1. utast jelenti.

ulesek=[]
for i in range(48): # az i index az i+1. ülést jelenti
    ulesek.append(None)

for i in utasok:
    ulesek[helyek[i]-1]=i

kitxt = open("kihol.txt","w")

for i in range(48):
    kitxt.write(f'{i+1}. ülés: ')
    if ulesek[i]==None:
        kitxt.write(f'üres\n')
    else:
        kitxt.write(f'{ulesek[i]+1}. utas\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

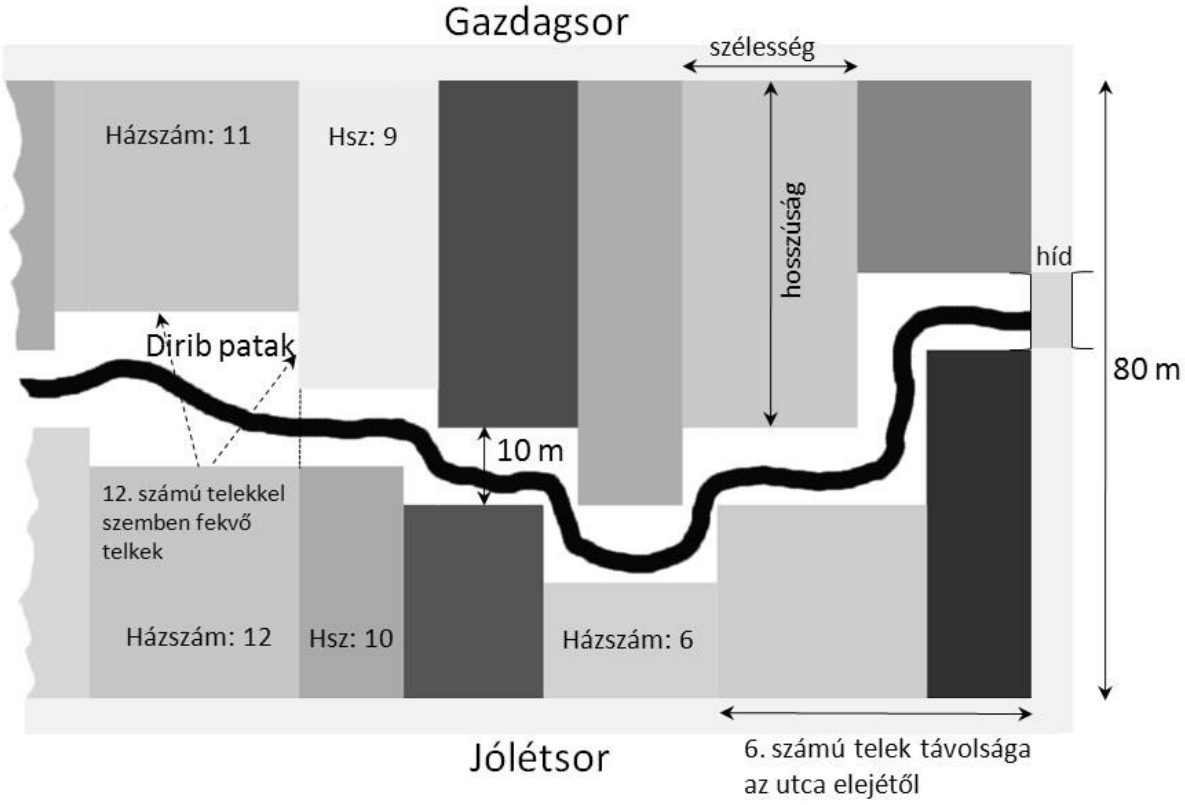
```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2010maj\EmeltInfo2010maj.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

2. feladat

Az utolsó jegyvásárló ülésének száma: 29, a beutazott távolság: 50 km.

3. feladat

A teljes utat végigutazók listája: 67 71 95

4. feladat

A társaság bevétele a jegyárakból: 46210 Ft.

5. feladat

A végállomás előtti utolsó megálló: 165 km-nél volt.
A megállóban 2 felszálló és 5 leszálló volt.

6. feladat

A busz a kiinduló állomás és a célállomás között 24 helyen állt meg.

7. feladat

Kérem az út egy tetszőleges km-ét (km<=172): 56

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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!

```
"""
2010. május idegennyelvű: Telek
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("telkek.txt")
tdb = int(betxt.readline().strip())

gsor,jsor=[], []

for i in range(tdb):
    darsor=betxt.readline().strip().split()
    adatsor=list(map(int,darsor))
    # 0: hsz, 1: szélesség, 2: hossz
    if adatsor[0]%2==1:
        gsor.append(adatsor)
    else:
        jsor.append(adatsor)

# Házszám szerinti rendezések:
gsor.sort(key=lambda elem:elem[0])
jsor.sort(key=lambda elem:elem[0])

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")

print("\n2. feladat\n")

# Mivel a telek téglalap, elég pl. Jólétsor hosszát meghatározni.
jhossz=0
for telek in jsor:
    jhossz+=telek[1]

print(f'A két utca körbejárásához {2*(jhossz+80)} métert kell gyalogolni.')


print("\n3. feladat\n")

teljesdb=0
for telek in jsor:
    if telek[1]<=20:
        teljesdb+=1

print(f'A teljes utcafront beépítések száma Jólétsoron: {teljesdb}')


print("\n4. feladat\n")

hszmin,tmin=gsor[0][0],gsor[0][1]*gsor[0][2]
hszmax,tmax=hszmin,tmin

for telek in gsor[1:]:
    ter=telek[1]*telek[2]
    if ter<tmin:
        hszmin=telek[0]
        tmin=ter
    elif ter>tmax:
        hszmax=telek[0]
        tmax=ter

print(f'A legnagyobb területű telek házszáma {hszmax}, területe {tmax} nm.')
print(f'A legkisebb területű telek házszáma {hszmin}, területe {tmin} nm.')
print(f'{(abs(hszmax-hszmin)-2)//2} darab telek van közöttük.')
```

```
print("\n5. feladat\n")

def adoszamito(szel, hossz):
    ter=szel*hossz
    if ter<=700:
        ado=ter*51
    elif ter<=1000:
        ado=700*51+(ter-700)*39
    else:
        ado=700*51+300*39+200
    if szel<=15 or hossz<=25:
        ado=int(ado*0.8)
    m=ado%100
    ado-=m
    if m>=50:
        ado+=100
    return ado

adobev=0
for telek in gsor:
    adobev+=adoszamito(telek[1],telek[2])

print(f'Gazdagsoron az önkormányzat {adobev} Fabatka telekadó bevételre számíthat.')


print("\n6. feladat\n")

print("Jólétsor utolsó három telke és távolságuk Jólétsor elejétől:")

tav=jhossz
for i in range(len(jsor)-1,len(jsor)-4,-1):
    tav-=jsor[i][1]
    print(f'{jsor[i][0]} {tav} m ')


print("\n7. feladat\n")

"""
    a_______b
      c_______d
A két intervallumnak akkor és csakis akkor van közös pontja, ha c<=b és d>=a
"""

def szemben(a,b,c,d):
    return c<=b and d>=a

# A telkek távolságai az utcák elejéről:
gtav,jtav=0,0
gtavok,jtavok=[],[]

for i in range(len(gsor)):
    gtavok.append(gtav)
    gtav+=gsor[i][1]
    jtavok.append(jtav)
    jtav+=jsor[i][1]

kitxt = open("joletsor.csv","w")

for i,jtelek in enumerate(jsor):
    gh=0
    for j,gtelek in enumerate(gsor):
        if szemben(jtavok[i],jtavok[i]+jtelek[1],gtavok[j],gtavok[j]+gtelek[1]):
            gh=max(gh,gtelek[2])
    kitxt.write(f'{jtelek[0]};{jtelek[1]};{80-10-gh}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2010mid/EmeltInfo2010mid.py ==================

1. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

2. feladat

A két utca körbejárásához 920 métert kell gyalogolni.

3. feladat

A teljes utcafront beépítések száma Jólétsoron: 8

4. feladat

A legnagyobb területű telek házszáma 27, területe 1225 nm.
A legkisebb területű telek házszáma 15, területe 600 nm.
5 darab telek van közöttük.

5. feladat

Gazdagsoron az önkormányzat 629100 Fabatka telekadó bevételre számíthat.

6. feladat

Jólétsor utolsó három telke és távolságuk Jólétsor elejétől:
32 360 m
30 325 m
28 300 m

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.


A befejezéshez nyomd meg az ENTER billentyűt!
```

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

```
"""
2010. október: Anagramma
@author Klemand66
"""

print("\n1. feladat\n")

szoveg=input("Kérek egy szöveget: ")

karakterek=[]
for kar in szoveg:
    if kar not in karakterek:
        karakterek.append(kar)

print(f'A megadott szövegben {len(karakterek)} különböző karakter található: {" ".join(karakterek)}')



print("\n2. feladat\n")

betxt = open("szotar.txt")

szavak=[]
for sor in betxt:
    szavak.append(sor.strip())

betxt.close()
print("A beolvasás és a fájl lezárása sikeresen befejeződött.")



print("\n3. feladat\n")

def beturendezo(szoveg):
    karlista=list(szoveg)
    karlista.sort()
    return "".join(karlista)

kitxt = open("abc.txt","w")

for szo in szavak:
    kitxt.write(f'{beturendezo(szo)}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")



print("\n4. feladat\n")

szo1=input("Kérek egy szót: ")
szo2=input("Kérek egy másik szót: ")

if beturendezo(szo1)==beturendezo(szo2):
    print("Anagramma")
else:
    print("Nem anagramma")



print("\n5. feladat\n")

ujszo=input("Kérek egy szót: ")
ujszo=beturendezo(ujszo)

anagrammak=[]
for szo in szavak:
    if beturendezo(szo)==ujszo:
        anagrammak.append(szo)

if len(anagrammak)>0:
    print("A szó anagrammái a szótárban:")
    for anagramma in anagrammak:
        print(anagramma)
else:
    print("Nincs a szótárban anagramma.")
```

```
print("\n6. feladat\n")

vezerabck,anagrammalistak=[],[]
for szo in szavak:
    szoabc=beturendezo(szo)
    if szoabc in vezerabck:
        aktindex=vezerabck.index(szoabc)
        anagrammalistak[aktindex].append(szo)
    else:
        vezerabck.append(szoabc)
        anagrammalistak.append([szo]) # itt nyitjuk az új anagrammalistát

hosszak=list(map(len,vezerabck))
maxhossz=max(hosszak)
leghosszabbak=list(filter(lambda i: len(vezerabck[i])==maxhossz,range(len(vezerabck))))

for i in leghosszabbak:
    for anagramma in anagrammalistak[i]:
        print(anagramma)
    print()



print("\n7. feladat\n")

anagrammalistak2d=list(map(lambda i:[anagrammalistak[i],len(vezerabck[i])],range(len(vezerabck))))

anagrammalistak2d.sort(key=lambda elem:elem[1])

kitxt = open("rendezve.txt","w")

hossz=min(hosszak)
for elem in anagrammalistak2d:
    if elem[1]>hossz:
        kitxt.write("\n")
        hossz=elem[1]
    kitxt.write(f'{" ".join(elem[0])}\n')


kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2010okt/EmeltInfo2010okt.py ==================

1. feladat

Kérek egy szöveget: informatika erettsegi
A megadott szövegben 13 különböző karakter található: i n f o r m a t k   e s g

2. feladat

A beolvasás és a fájl lezárása sikeresen befejeződött.

3. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.


4. feladat

Kérek egy szót: kalap
Kérek egy másik szót: lapka
Anagramma

5. feladat

Kérek egy szót: rekeszt
A szó anagrammái a szótárban:
kereszt

6. feladat

hajnalokat
hatoljanak

kalandtura


7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.


A befejezéshez nyomd meg az ENTER billentyűt!
```

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

...
```

```
"""
2011. május: Szójáték
@author Klemand66
"""

print("\n1. feladat\n")

def mghszamlalo(szoveg):
    mghk="aeiou"
    mghlista=list(filter(lambda betu:betu in mghk,szoveg))
    return len(mghlista)

szo=input("Kérek egy szót, mely csak az angol abc kisbetűit tartalmazza: ")

if mghszamlalo(szo)>0:
    print("Van benne magánhangzó.")
else:
    print("Nincs benne magánhangzó.")



print("\n2. feladat\n")

betxt = open("szoveg.txt")

"""
Csak beolvassuk egyenként a sorokat, és tárolás nélkül dolgozzuk fel.
Így most nem alkalmazható a max(), map(), filter() függvények egyike sem,
a klasszikus programozási tételeket kell használni.
"""

maxhossz,maxszo=0,None
for sor in betxt:
    aktszo=sor.strip()
    if len(aktszo)>maxhossz:
        maxhossz=len(aktszo)
        maxszo=aktszo

betxt.close()
print(f'A leghosszabb szó: {maxszo}, hossza: {maxhossz} karakter.')



print("\n3. feladat\n")

print("A több magánhangzót tartalmazó szavak:")

betxt = open("szoveg.txt")

db,dbtobb=0,0

for sor in betxt:
    db+=1
    aktszo=sor.strip()
    if len(aktszo)<2*mghszamlalo(aktszo):
        print(aktszo,end=" ")
        dbtobb+=1
        if dbtobb%12==0:
            print()

betxt.close()

print(f'\n{dbtobb}/{db} : {100*dbtobb/db:.2f}%')
```

```
print("\n4. feladat\n")

betxt = open("szoveg.txt")

szavak5=[]
for sor in betxt:
    aktszo=sor.strip()
    if len(aktszo)==5:
        szavak5.append(aktszo)

betxt.close()

fok=input("Kérek egy 3 karakteres szórészletet (pl. isz vagy obo): ")

print(f'A megadott {fok} létrafok létraszavai: ')

for szo in szavak5:
    if szo[1:-1]==fok:
        print(szo,end=" ")
print()



print("\n5. feladat\n")

fokok,letraszolistak=[],[]
for szo in szavak5:
    aktfok=szo[1:-1]
    if aktfok in fokok:
        aktindex=fokok.index(aktfok)
        letraszolistak[aktindex].append(szo)
    else:
        fokok.append(aktfok)
        letraszolistak.append([szo]) # itt nyitjuk az új létraszólistát

kitxt = open("letra.txt","w")

for lista in letraszolistak:
    if len(lista)>1:
        for szo in lista:
            kitxt.write(f'{szo}\n')
        kitxt.write("\n")

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2011maj/EmeltInfo2011maj.py ==================

1. feladat

Kérek egy szót, mely csak az angol abc kisbetűit tartalmazza: nap
Van benne magánhangzó.

2. feladat

A leghosszabb szó: angyalszobrokat, hossza: 15 karakter.

3. feladat

A több magánhangzót tartalmazó szavak:
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%

4. feladat

Kérek egy 3 karakteres szórészletet (pl. isz vagy obo): egy
A megadott egy létrafok létraszavai:
hegye hegyi megye tegye vegye

5. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!


A letra.txt szövegfájl

addig                   arcok                   rakta                   pokol
eddig                   arcon
                        arcot                   balra                   borok
adtak                                           falra                   boros
adtam                   arcuk                                           forog
                        arcul                   barom                   korom
agyag                                           karok                   koron
ugyan                   babot                   karom                   poros
                        habos                   karon                   sorom
agyba                   rabok                   marok                   soron
egybe                   rabom                   maros                   torok
                        rabon
akkor                   rabot                   batyu                   borul
ekkor                   zabot                   matyi                   poruk

aludj                   bajba                   benne                   budai
aludt                   hajba                   lenne                   cudar
                        lajbi                   lenni                   ludak
amely                                           menni                   rudat
emeli                   bajod                   tenni
emelt                   bajom                   venni                   cecey
                        bajos                                           pecek
annak                   hajol                   beteg
onnan                   lajos                   hetet                   cicus
                        vajon                   vetem                   vicus
annyi                   zajos                                           ...
ennyi                                           bokor
                        bakta                   fokot
```

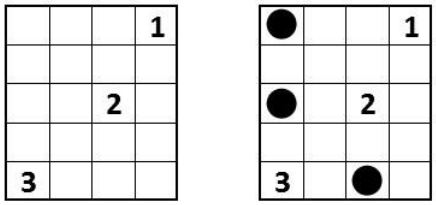
2011. május, idegennyelvű: Rejtvény

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!

```
"""
2011. május idegennyelvű: Rejtvény
@author Klemand66
"""

print("\n1. feladat\n")

ts=int(input("Kérem a világítótorony pozícióját. Sorának száma: "))
to=int(input("Oszlopának száma: "))
hdb=int(input("Kérem a toronyból látható hajók számát: "))
if hdb>3:
    print("Nehéz torony.")


print("\n2. feladat\n")

print(f'A megadott {ts},{to} helyzetű torony körüli tiltott helyek:')

for s in range(1,11):
    for o in range (1,11):
        if (abs(s-ts)==1 and o in [to-1,to,to+1]) or ((s==ts and o in [to-1,to+1])):
            print(f'{s},{o}')


print("\n3. feladat\n")

"""
A könnyebb feldolgozás kedvéért a 10x10-es táblázatokat sorfolytonos listában tároljuk.
A transzformációk:
sor=index//10 +1, oszlop=index%10 +1
index=(sor-1)*10+(oszlop-1)
"""

betxt = open("feladvany.txt")

feladvany=[]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=list(map(int,darsor))
    feladvany+=adatsor

betxt.close()

betxt = open("megoldas.txt")

bekuldok,megoldasok=[],[]

mdb=int(betxt.readline().strip())

for i in range(mdb):
    bekuldok.append(betxt.readline().strip())
    megoldas=[]
    for j in range(10):
        darsor=betxt.readline().strip().split()
        adatsor=list(map(int,darsor))
        megoldas+=adatsor
    megoldasok.append(megoldas)

betxt.close()

jomegoldasok=[]
for i in range(len(bekuldok)):
    jomegoldasok.append(True)

print("Nem erre a feladványra küldtek megoldást:")
db=0
for i,megoldas in enumerate(megoldasok):
    alap=list(map(lambda elem:elem%11,megoldas)) # lenullázzuk a hajókat
    if alap!=feladvany:
        print(bekuldok[i])
        db+=1
        jomegoldasok[i]=False

if db>0:
    print(f'{db} beküldés nem erre a hétre érkezett.')
else:
    print("Mindegyik beküldés erre a feladványra érkezett.")
```

```
print("\n4. feladat\n")

print("Nem jó a hajók száma:")
db=0
for i,megoldas in enumerate(megoldasok):
    if not jomegoldasok[i]:
        continue # aki nem erre a hétre küldte, azt csak átugorjuk
    if megoldas.count(11)!=12:
        print(bekuldok[i])
        db+=1
        jomegoldasok[i]=False

if db>0:
    print(f'{db} fennmaradt beküldésben nem volt jó a hajók száma.')
else:
    print("Mindegyik fennmaradt beküldésben megfelelő számú hajó volt.")


print("\n5. feladat\n")

def tiltott(objind,moind):
    objs, objo, mos, moo  = objind//10 +1, objind%10 +1, moind//10 +1, moind%10 +1
    return (abs(mos-objs)==1 and moo in [objo-1,objo,objo+1]) or (mos==objs and moo in [objo-1,objo+1])

print("Nem szabályos a hajók elhelyezése:")
db=0
for i,megoldas in enumerate(megoldasok):
    if not jomegoldasok[i]:
        continue # a korábban kiesetteket csak átugorjuk
    szabalytalan=False
    objektumindexek=list(filter(lambda j:megoldas[j]>0,range(len(megoldas))))
    for obji in objektumindexek:
        for moi in range(len(megoldas)):
            if tiltott(obji,moi):
                if moi in objektumindexek:
                    db+=1
                    print(bekuldok[i])
                    jomegoldasok[i]=False
                    szabalytalan=True
            if szabalytalan:
                break
        if szabalytalan:
            break

if db>0:
    print(f'{db} fennmaradt beküldésben volt szabálytalan a hajók elrendezése.')
else:
    print("Mindegyik fennmaradt beküldésben szabályos a hajók elrendezése.")
```

```
print("\n6. feladat\n")

print("Nem annyi hajó látszik:")
db=0
for i,megoldas in enumerate(megoldasok):
    if not jomegoldasok[i]:
        continue
    rossz=False
    toronyindexek=list(filter(lambda j:feladvany[j]>0,range(len(feladvany))))
    tornyok=list(filter(lambda elem:elem>0,feladvany))
    hajoindexek=list(filter(lambda j:megoldas[j]==11,range(len(megoldas))))
    # A torony látja a hajót, ha az indexeik első vagy második jegye megegyezik.
    for j,ti in enumerate(toronyindexek):
        hdb=0
        for hi in hajoindexek:
            if ti//10==hi//10 or ti%10==hi%10:
                hdb+=1
        if hdb!=tornyok[j]:
            db+=1
            print(bekuldok[i])
            jomegoldasok[i]=False
            rossz=True
        if rossz:
                break

if db>0:
    print(f'{db} fennmaradt beküldésben volt rossz a hajók elrendezése.')
else:
    print("Mindegyik fennmaradt beküldésben jó a hajók elrendezése.")

print(f'\nEzen a héten {jomegoldasok.count(True)} helyes megoldás érkezett a feladványra.')
print("A helyes megoldást beküldők névsora:")

for i,bekuldo in enumerate(bekuldok):
    if jomegoldasok[i]:
        print(bekuldo)


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2011mid/EmeltInfo2011mid.py ==================

1. feladat

Kérem a világítótorony pozícióját. Sorának száma: 9
Oszlopának száma: 1
Kérem a toronyból látható hajók számát: 4
Nehéz torony.

2. feladat

A megadott 9,1 helyzetű torony körüli tiltott helyek:
8,1
8,2
9,2
10,1
10,2

3. feladat

Nem erre a feladványra küldtek megoldást:
Ildefonz
Ozor
2 beküldés nem erre a hétre érkezett.

4. feladat

Nem jó a hajók száma:
Absolon
Verner
2 fennmaradt beküldésben nem volt jó a hajók száma.

5. feladat

Nem szabályos a hajók elhelyezése:
Selton
Meliton
2 fennmaradt beküldésben volt szabálytalan a hajók elrendezése.

6. feladat

Nem annyi hajó látszik:
Agaton
1 fennmaradt beküldésben volt rossz a hajók elrendezése.

Ezen a héten 3 helyes megoldás érkezett a feladványra.
A helyes megoldást beküldők névsora:
Kasztor
Folkus
Bazil

A befejezéshez nyomd meg az ENTER billentyűt!
```

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!

```
"""
2011. október: Pitypang
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("honapok.txt")

honapok=[]
for i in range(12):
    honap=[(betxt.readline().strip())]          # a hónap neve, nem használjuk
    honap.append(int(betxt.readline().strip())) # napok száma
    honap.append(int(betxt.readline().strip())) # az év hónapkezdő napja
    honapok.append(honap)

betxt.close()

nyk,oszk=121,244 # a nyári és őszi időszak kezdőnapja

betxt = open("pitypang.txt")

fdb=int(betxt.readline().strip())

foglalasok=[]
for i in range(fdb):
    darsor=betxt.readline().strip().split()
    foglalas=[int(darsor[0])]       # sorszám, az index+1 alapján is tudnánk
    foglalas.append(int(darsor[1])) # szobaszám
    foglalas.append(int(darsor[2])) # érkezés napja
    foglalas.append(int(darsor[3])) # távozás napja
    foglalas.append(int(darsor[4])) # vendégek száma
    foglalas.append(int(darsor[5])) # reggeli (1: igen, 1: nem)
    foglalas.append(darsor[6])      # a foglalás azonosítója
    foglalasok.append(foglalas)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")


print("\n2. feladat\n")

print("A leghosszabb tartózkodás(ok)\n")

print("A map() függvénnyel:")

"""
A map() függvény segítségével egy többdimenziós lista
oszlopai alapján is készíthetünk listát:
"""
ejszdbk=list(map(lambda foglalas:foglalas[3]-foglalas[2],foglalasok))
lhtart=max(ejszdbk)
"""
A filter() függvénnyel könnyen megadhatjuk az összes leghosszabb tartózkodást is:
"""
lhindexek=list(filter(lambda i:ejszdbk[i]==lhtart,range(fdb)))
for i in lhindexek:
    print(f'{foglalasok[i][6]} ({foglalasok[i][2]}) - {lhtart}')

print("\nA max() függvénnyel:")
"""
Vagy közvetlenül alkalmazhatjuk most is a max() függvényt is
a lambda függvénnyel megadott kulcs segítségével:
"""
lhfoglalas=max(foglalasok, key=lambda foglalas:foglalas[3]-foglalas[2])
"""
Az lhfoglalás nem a leghosszabb tartózkodás idejét,
hanem az ehhez tartozó teljes rekordot (foglalást) tartalmazza
"""
print(f'{lhfoglalas[6]} ({lhfoglalas[2]}) - {lhfoglalas[3]-lhfoglalas[2]}')
```

```
print("\n3. feladat\n")

kitxt = open("bevetel.txt","w")

osszbev=0
for i,foglalas in enumerate(foglalasok):
    if foglalas[2]>=oszk:
        ear=8000
    elif foglalas[2]<nyk:
        ear=9000
    else:
        ear=10000
    bevetel=ejszdbk[i]*ear
    if foglalas[4]>2:
        bevetel+=ejszdbk[i]*2000
    if foglalas[5]==1:
        bevetel+=foglalas[4]*ejszdbk[i]*1100
    kitxt.write(f'{foglalas[0]}:{bevetel}\n')
    osszbev+=bevetel

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

print(f'A szálloda teljes évi bevétele {osszbev:,} Ft.')


print("\n4. feladat\n")

def honaphatarozo(evnapja): # a feladat szerint nincs szökőév
    honaplistak=list(map(lambda honap: range(honap[2],honap[2]+honap[1]), honapok))
    bennevan=list(map(lambda honap: evnapja in honap, honaplistak))
    return bennevan.index(True)

vejdbk=[]
for i in range(12):
    vejdbk.append(0)

for foglalas in foglalasok:
    for i in range(foglalas[2],foglalas[3]):
        aktindex=honaphatarozo(i)
        vejdbk[aktindex]+=foglalas[4]

print("Statisztika a havi vendégéjszakákról:")
for i,n in enumerate(vejdbk):
    print(f'{i+1:2}: {n:4} vendégéj')


print("\n5. feladat\n")

print("Kérem a foglalás adatait! Az időszak teljes egészében az idei évre essen!")

ujerk=int(input("Kérem az érkezés időpontjának sorszámát: "))
ujejdb=int(input("Kérem az eltölteni kívánt vendégéjszakák számát: "))

print("A megadott időszakban a következő szobák szabadok:")
szobadb=27
szabaddb=0

"""
    e ------o t
      ue -------o ut
Ütközés: ue<t és ut>e
"""

for i in range(1,szobadb+1):
    szabad=True
    for foglalas in foglalasok:
        if foglalas[1]==i:
            if (ujerk<foglalas[3] and ujerk+ujejdb>foglalas[2]):
                szabad=False
                break
    if szabad:
        print(i,end=" ")
        szabaddb+=1

print(f'\nA szabad szobák száma: {szabaddb}')

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2011okt/EmeltInfo2011okt.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A leghosszabb tartózkodás(ok)

A map() függvénnyel:
Meszaros_Agnes (121) - 17

A max() függvénnyel:
Meszaros_Agnes (121) - 17

3. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A szálloda teljes évi bevétele 73,053,900 Ft.

4. feladat

Statisztika a havi vendégéjszakákról:
 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

5. feladat

Kérem a foglalás adatait! Az időszak teljes egészében az idei évre essen!
Kérem az érkezés időpontjának sorszámát: 90
Kérem az eltölteni kívánt vendégéjszakák 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 befejezéshez nyomd meg az ENTER billentyűt!
```

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
...
```

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!

```
"""
2012. május: Futár
@author Klemand66
"""


print("\n1. feladat\n")

betxt = open("tavok.txt")

utak=[]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=list(map(int,darsor))
    # 0: nap (1-7), 1: fuvarszám (1-40), 2: km (<=30)
    utak.append(adatsor)
betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

utak.sort(key=lambda x:x[1]) # először a mellékszempont (fuvarszám) szerint rendezünk
utak.sort(key=lambda x:x[0]) # másodszor pedig a főszempont (nap) szerint

print(f'A hét első útja {utak[0][2]} km volt.')



print("\n3. feladat\n")

print(f'A hét utolsó útja {utak[-1][2]} km volt.')



print("\n4. feladat\n")

naplista=list(map(lambda x: x[0],utak))
sznapok=list(filter(lambda i: i not in naplista, range(1,8)))
sznapokstr=list(map(str,sznapok))
print(f'A futár a következő napokon nem dolgozott: {". nap, ".join(sznapokstr)}. nap')



print("\n5. feladat\n")

maxfuvarut=max(utak,key=lambda x: x[1])
print(f'A futárnak a hét {maxfuvarut[0]}. napján volt a legtöbb fuvarja: {maxfuvarut[1]}')

# Szorgalmi feladat: az összes nap megkeresése, amikor maximális számú fuvarja volt:
maxfuvar=maxfuvarut[1]
maxfuvarutak=list(filter(lambda x: x[1]==maxfuvar, utak))

print("\nSzűréssel az összes nap, amikor maximális számú fuvarja volt: ")

for ut in maxfuvarutak:
    print(f'{ut[0]}. napon {ut[1]} fuvar')



print("\n6. feladat\n")

napikmek=[]
for i in range(7): # a listaindex 0-6-ig terjed
    napikmek.append(0) # kezdőértékek beállítása

for ut in utak:
    aktnap=ut[0]
    napikmek[aktnap-1]+=ut[2] # érték szerinti indexelés

for i in range(1,8): # a napsorszám 1-7-ig terjed
        print(f'{i}. nap: {napikmek[i-1]} km')
```

```
print("\n7. feladat\n")

def dij(km):
    tavhatarok=[0,2,5,10,20,30]
    dijak=[500,700,900,1400,2000]
    bennevan=list(map(lambda i: tavhatarok[i-1]<km and km<=tavhatarok[i], range(1,len(tavhatarok))))
    aktindex=bennevan.index(True)
    return dijak[aktindex]

"""
Most talán egyszerűbb lenne az elif-es algoritmus, de sok sávnál már nem elegáns.

def dij(km):
    if km<=2:
        return 500
    elif km<=5:
        return 700
    elif km<=10:
        return 900
    elif km<=20:
        return 1400
    else:
        return 2000
"""

tav=int(input("Kérem a megtett táv hosszát km-ben (max 30): "))
print(f'A megtett távért {dij(tav)} Ft díjazás jár.')



print("\n8. feladat\n")

kitxt = open("dijazas.txt","w")

for ut in utak:
   kitxt.write(f'{ut[0]}. nap {ut[1]:2}. út: {dij(ut[2]):4} Ft\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n9. feladat\n")

hetidij=0
for ut in utak:
    hetidij+=dij(ut[2])

print(f'A futárnak a heti munkájáért {hetidij} Ft díjazás jár.')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2012maj/EmeltInfo2012maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A hét első útja 3 km volt.

3. feladat

A hét utolsó útja 25 km volt.

4. feladat

A futár a következő napokon nem dolgozott: 2. nap, 6. nap

5. feladat

A futárnak a hét 5. napján volt a legtöbb fuvarja: 21

Szűréssel az összes nap, amikor maximális számú fuvarja volt:
5. napon 21 fuvar

6. feladat

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

7. feladat

Kérem a megtett táv hosszát km-ben (max 30): 11
A megtett távért 1400 Ft díjazás jár.

8. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

9. feladat

A futárnak a heti munkájáért 48500 Ft díjazás jár.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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
```

```
"""
2012. május idegennyelvű: Törtek
@author Klemand66
"""

print("\n1. feladat\n")

sz=int(input("Kérem a tört számlálóját, egy pozitív, legfeljebb kétjegyű egész számot! "))
n=int(input("Kérem a tört nevezőjét, egy pozitív, legfeljebb kétjegyű egész számot! "))

m=sz%n
if m==0:
    print(f'A megadott tört egész szám: {sz}/{n} = {sz//n}')
else:
    print("Nem egész.")



print("\n2. feladat\n")

def lnko(a,b):
    if a==b:
        return a
    if a<b:
        return lnko(a,b-a)
    if a>b:
        return lnko(a-b,b)

print(f'Az lnko függvény:\n\n\
def lnko(a,b):\n\
    if a==b:\n\
        return a\n\
    if a<b:\n\
        return lnko(a,b-a)\n\
    if a>b:\n\
        return lnko(a-b,b)')



print("\n3. feladat\n")

def formazo(sz,n):
    if sz%n == 0:
        return f'{sz//n}'
    else:
        return f'{sz//lnko(sz,n)}/{n//lnko(sz,n)}'

print(f'{sz}/{n} = {formazo(sz,n)}')



print("\n4. feladat\n")

sz2=int(input("Kérem egy másik tört számlálóját, egy pozitív, legfeljebb kétjegyű egész számot! "))
n2=int(input("Kérem a másik tört nevezőjét, egy pozitív, legfeljebb kétjegyű egész számot! "))

print(f'{sz}/{n} * {sz2}/{n2} = {sz*sz2}/{n*n2}  =  {formazo(sz*sz2,n*n2)}')



print("\n5. feladat\n")

def lkkt(a,b):
    return a*b//lnko(a,b)

print(f'Az lkkt függvény:\n\n\
def lkkt(a,b):\n\
    return a*b//lnko(a,b)')
```

```
print("\n6. feladat\n")


print(f'{sz}/{n} + {sz2}/{n2} = \
{sz*lkkt(n,n2)//n}/{lkkt(n,n2)} + {sz2*lkkt(n,n2)//n2}/{lkkt(n,n2)} = \
{sz*lkkt(n,n2)//n + sz2*lkkt(n,n2)//n2}/{lkkt(n,n2)} = \
{formazo(sz*lkkt(n,n2)//n + sz2*lkkt(n,n2)//n2,lkkt(n,n2))}')



print("\n7. feladat\n")

betxt = open("adat.txt")
kitxt = open("eredmeny.txt","w")

for sor in betxt:
    darsor=sor.strip().split()
    sz,n,sz2,n2,muv=int(darsor[0]),int(darsor[1]),int(darsor[2]),int(darsor[3]),darsor[4]

    if muv=="*":
        kitxt.write(f'{sz}/{n} * {sz2}/{n2} = {sz*sz2}/{n*n2}  =  {formazo(sz*sz2,n*n2)}\n')
    else:
        kitxt.write(f'{sz}/{n} + {sz2}/{n2} = \
{sz*lkkt(n,n2)//n}/{lkkt(n,n2)} + {sz2*lkkt(n,n2)//n2}/{lkkt(n,n2)} = \
{sz*lkkt(n,n2)//n + sz2*lkkt(n,n2)//n2}/{lkkt(n,n2)} = \
{formazo(sz*lkkt(n,n2)//n + sz2*lkkt(n,n2)//n2,lkkt(n,n2))}\n')

betxt.close()
kitxt.close()
print("A beolvasás, feldolgozás, kiíratás és a szövegfájlok lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2012mid/EmeltInfo2012mid.py ==================

1. feladat

Kérem a tört számlálóját, egy pozitív, legfeljebb kétjegyű egész számot! 24
Kérem a tört nevezőjét, egy pozitív, legfeljebb kétjegyű egész számot! 32
Nem egész.

2. feladat

Az lnko függvény:

def lnko(a,b):
    if a==b:
        return a
    if a<b:
        return lnko(a,b-a)
    if a>b:
        return lnko(a-b,b)

3. feladat

24/32 = 3/4

4. feladat

Kérem egy másik tört számlálóját, egy pozitív, legfeljebb kétjegyű egész számot! 8
Kérem a másik tört nevezőjét, egy pozitív, legfeljebb kétjegyű egész számot! 3
24/32 * 8/3 = 192/96 = 2

5. feladat

Az lkkt függvény:

def lkkt(a,b):
    return a*b//lnko(a,b)

6. feladat

24/32 + 8/3 = 72/96 + 256/96 = 328/96 = 41/12

7. feladat

A beolvasás, feldolgozás, kiíratás és a szövegfájlok lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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
12/45 + 10/6 = 24/90 + 150/90 = 174/90 = 29/15
50/12 * 4/12 = 200/144 = 25/18
20/2 * 50/2 = 1000/4 = 250
12/27 + 20/2 = 24/54 + 540/54 = 564/54 = 94/9
30/15 * 2/75 = 60/1125 = 4/75
1/30 * 8/9 = 8/270 = 4/135
12/4 * 20/30 = 240/120 = 2
4/15 + 12/30 = 8/30 + 12/30 = 20/30 = 2/3
20/75 * 50/75 = 1000/5625 = 8/45
3/45 + 45/10 = 6/90 + 405/90 = 411/90 = 137/30
20/4 + 30/6 = 60/12 + 60/12 = 120/12 = 10
12/20 + 25/15 = 36/60 + 100/60 = 136/60 = 34/15
20/6 * 2/18 = 40/108 = 10/27
8/75 * 3/50 = 24/3750 = 4/625
10/75 * 5/10 = 50/750 = 1/15
8/10 + 75/6 = 24/30 + 375/30 = 399/30 = 133/10
9/2 * 10/30 = 90/60 = 3/2
```

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
```

```
"""
2012. október: Szín-kép
@author Klemand66
"""

print("\n1. feladat\n")

"""
A könnyebb feldolgozás kedvéért a képpontokat nem 50x50-es táblázatban,
hanem a fájlnak megfelelő sorfolytonos listában tároljuk.
A transzformációk:
sor=index//50 +1, oszlop=index%50 +1
index=(sor-1)*50+(oszlop-1)
"""

betxt = open("kep.txt")

szinek=[]
for sor in betxt:
    szinek.append(sor.strip())

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.\n")

# Szorgalmi feladat: a kép ábrázolása:

def szinjel(kod):
    kodok=["255 0 0","0 255 0","0 0 255","255 255 0","255 0 255","0 255 255","0 0 0","255 255 255"]
    jelek=["R","G","B","Y","M","C","B","W"]
    if kod in kodok:
        aktindex=kodok.index(kod)
        return jelek[aktindex]
    else:
        return "-"

for i,szin in enumerate(szinek):
    print(szinjel(szin),end="")
    if (i+1)%50==0:
        print()



print("\n2. feladat\n")

R=input("Kérem egy szín RGB-kódját. A vörös összetevő értéke (0<=R<=255): ")
G=input("A zöld összetevő értéke (0<=G<=255): ")
B=input("A kék összetevő értéke (0<=B<=255): ")

aktszin=f'{R} {G} {B}'

if aktszin in szinek:
    aktindex=szinek.index(aktszin)
    print(f'A megadott {aktszin} RGB-kódú szín megtalálható \
a kép {aktindex//50 +1}. sorának {aktindex%50 +1}. képpontjában. ')
else:
    print(f'A megadott {aktszin} RGB-kódú szín nem található meg a képen.')



print("\n3. feladat\n")

s,o=35,8
aktindex=(s-1)*50 +o-1
aktszin=szinek[aktindex]

aktsor=list(filter(lambda i: i//50+1==s ,range(len(szinek))))
aktszinek=list(map(lambda i: szinek[i] , aktsor))
sdb=aktszinek.count(aktszin)

aktoszlop=list(filter(lambda i: i%50+1==o ,range(len(szinek))))
aktszinek=list(map(lambda i: szinek[i] , aktoszlop))
odb=aktszinek.count(aktszin)

print(f'Sorban: {sdb} Oszlopban: {odb}')
```

```
print("\n4. feladat\n")

vdb=szinek.count("255 0 0")
zdb=szinek.count("0 255 0")
kdb=szinek.count("0 0 255")
maxdb=max(vdb,zdb,kdb)

# Ha az összes legtöbbször előforduló színt ki akarom írni:
print("A legtöbbször előforduló színek: ", end="")
if vdb==maxdb:
    print("vörös ",end="")
if zdb==maxdb:
    print("zöld ",end="")
if kdb==maxdb:
    print("kék ",end="")
print()

# Ha csak az egyik legtöbbször előforduló színt akarom kiírni:
print("Az egyik legtöbbször előforduló szín: ", end="")
if vdb==maxdb:
    print("vörös ",end="")
elif zdb==maxdb:
    print("zöld ",end="")
elif kdb==maxdb:
    print("kék ",end="")
print()



print("\n5. feladat\n")

keret=[1,2,3,48,49,50]

for i in range(len(szinek)):
    s,o=i//50+1, i%50+1
    if s in keret or o in keret:
        szinek[i]="0 0 0"

# Szorgalmi feladat: a kép ábrázolása

for i,szin in enumerate(szinek):
    print(szinjel(szin),end="")
    if (i+1)%50==0:
        print()



print("\n6. feladat\n")

kitxt = open("keretes.txt","w")

for szin in szinek:
    kitxt.write(f'{szin}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n7. feladat\n")

stl=list(filter(lambda i: szinek[i]=="255 255 0" ,range(len(szinek))))

print(f'Kezd: {stl[0]//50+1}, {stl[0]%50+1}')
print(f'Vége: {stl[-1]//50+1}, {stl[-1]%50+1}')
print(f'Képpontok száma: {len(stl)}')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2012okt/EmeltInfo2012okt.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
------------------------------MMMMMMMMMM---------
------------------------------MMMMMMMMMM---------
---------RRRRRRRRRR----------MMMMMMMMMM---------
---------RRRRRRRRRR------------------------------
---------RRRRRRRRRR------------------------------
---------RRRRRRRRRR------------------------------
---------RRRRRRRRRR------------------------------
---------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGG----------
---------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGG----------
---------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGG----------
---------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGG----------
---------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGG----------
---------RRRRRGGGGGGGGGGGGGGGBBBBBBBBBB----------
--------------GGGGGGGGGGGGGGGBBBBBBBBBB----------
--------------GGGGGGGGGGGGGGGBBBBBBBBBB----------
--------------GGGGGGGGGGGGGGGBBBBBBBBBB----------
--------------GGGGGGGGGGGGGGGBBBBBBBBBB----------
--------------GGGGGGGGGGGGGGGBBBBBBBBBB----------
------------------------------BBBBBBBBBB----------
------------------------------BBBBBBBBBB----------
------------------------------BBBBBBBBBB----------
------------------------------BBBBBBBBBB----------
------------------------------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY--------BBBBBBBBBB----------
-----------YYYYYYYYYY------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------
--------------------------------------------------

2. feladat

Kérem egy szín RGB-kódját. A vörös összetevő értéke (0<=R<=255): 255
A zöld összetevő értéke (0<=G<=255): 255
A kék összetevő értéke (0<=B<=255): 0
A megadott 255 255 0 RGB-kódú szín megtalálható a kép 31. sorának 12. képpontjában.
```

```
3. feladat

Sorban: 29 Oszlopban: 50

4. feladat

A legtöbbször előforduló színek: kék
Az egyik legtöbbször előforduló szín: kék

5. feladat

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBB--------------------------------------------BBB
BBB--------------------------------------------BBB
BBB--------------------------------------------BBB
BBB--------------------------------------------BBB
BBB---------------------------MMMMMMMMMMM------BBB
BBB---------------------------MMMMMMMMMMM------BBB
BBB------RRRRRRRRRRR----------MMMMMMMMMMM------BBB
BBB------RRRRRRRRRRR---------------------------BBB
BBB------RRRRRRRRRRR---------------------------BBB
BBB------RRRRRRRRRRR---------------------------BBB
BBB------RRRRRRRRRRR---------------------------BBB
BBB------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGGG-------BBB
BBB------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGGG-------BBB
BBB------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGGG-------BBB
BBB------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGGG-------BBB
BBB------RRRRRGGGGGGGGGGGGGGGGGGGGGGGGGG-------BBB
BBB------RRRRRGGGGGGGGGGGGGGGBBBBBBBBBBB-------BBB
BBB-----------GGGGGGGGGGGGGGGBBBBBBBBBBB-------BBB
BBB-----------GGGGGGGGGGGGGGGBBBBBBBBBBB-------BBB
BBB-----------GGGGGGGGGGGGGGGBBBBBBBBBBB-------BBB
BBB-----------GGGGGGGGGGGGGGGBBBBBBBBBBB-------BBB
BBB-----------GGGGGGGGGGGGGGGBBBBBBBBBBB-------BBB
BBB--------------------------BBBBBBBBBBB-------BBB
BBB--------------------------BBBBBBBBBBB-------BBB
BBB--------------------------BBBBBBBBBBB-------BBB
BBB--------------------------BBBBBBBBBBB-------BBB
BBB--------------------------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------BBBBBBBBBBB-------BBB
BBB--------YYYYYYYYYY--------------------------BBB
BBB--------------------------------------------BBB
BBB--------------------------------------------BBB
BBB--------------------------------------------BBB
BBB--------------------------------------------BBB
BBB--------------------------------------------BBB
BBB--------------------------------------------BBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
```

```
6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

7. feladat

Kezd: 31, 12
Vége: 41, 21
Képpontok száma: 110

A befejezéshez nyomd meg az ENTER billentyűt!
```

A keretes.txt szövegfájl

```
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               255 0 0
0 0 0                   200 96 64               200 96 64               200 96 64
0 0 0                   200 96 64               200 96 64               200 96 64
0 0 0                   200 96 64               200 96 64               200 96 64
0 0 0                   200 96 64               200 96 64               200 96 64
0 0 0                   ...                     200 96 64               200 96 64
0 0 0                   200 96 64               200 96 64               200 96 64
0 0 0                   0 0 0                   200 96 64               200 96 64
...                     0 0 0                   ...                     200 96 64
0 0 0                   0 0 0                   200 96 64               200 96 64
200 96 64               0 0 0                   255 0 255               200 96 64
200 96 64               0 0 0                   255 0 255               255 0 255
200 96 64               0 0 0                   255 0 255               255 0 255
200 96 64               200 96 64               255 0 255               255 0 255
200 96 64               200 96 64               255 0 255               255 0 255
200 96 64               200 96 64               255 0 255               255 0 255
200 96 64               200 96 64               255 0 255               255 0 255
200 96 64               200 96 64               255 0 255               255 0 255
200 96 64               200 96 64               255 0 255               255 0 255
200 96 64               200 96 64               255 0 255               255 0 255
200 96 64               200 96 64               255 0 255               255 0 255
200 96 64               200 96 64               200 96 64               255 0 255
200 96 64               200 96 64               200 96 64               200 96 64
200 96 64               200 96 64               200 96 64               200 96 64
200 96 64               200 96 64               200 96 64               200 96 64
200 96 64               200 96 64               200 96 64               200 96 64
200 96 64               200 96 64               200 96 64               200 96 64
200 96 64               200 96 64               0 0 0                   200 96 64
...                     200 96 64               0 0 0                   0 0 0
200 96 64               200 96 64               0 0 0                   0 0 0
0 0 0                   200 96 64               0 0 0                   0 0 0
0 0 0                   ...                     0 0 0                   0 0 0
0 0 0                   200 96 64               0 0 0                   0 0 0
0 0 0                   0 0 0                   200 96 64               0 0 0
0 0 0                   0 0 0                   200 96 64               200 96 64
200 96 64               0 0 0                   200 96 64               200 96 64
200 96 64               0 0 0                   200 96 64               200 96 64
200 96 64               0 0 0                   200 96 64               ...
```

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!

```
"""
2013. május: Választások
@author Klemand66
"""

print("\n1. feladat\n")

jogdb=12345

jeloltek=[]
betxt = open("szavazatok.txt")

for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[int(darsor[0]),int(darsor[1]),darsor[2]+" "+darsor[3],darsor[4]]
    # 0: körzet 1: szavazatok száma 2: név 3: támogató rövidítése
    jeloltek.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A helyhatósági választáson {len(jeloltek)} képviselőjelölt indult.')



print("\n3. feladat\n")

vnev=input("Kérem egy képviselő nevét. A vezetékneve: ")
unev=input("Az utóneve: ")
nev=vnev+" "+unev

aktjeloltek=list(filter(lambda jelolt:jelolt[2]==nev ,jeloltek))

if len(aktjeloltek)>0: # tudjuk, hogy most a szűrt lista max. egy elemű lesz
    print(f'{aktjeloltek[0][2]} {aktjeloltek[0][1]} szavazatot kapott.')
else:
    print("Ilyen nevű képviselőjelölt nem szerepel a nyilvántartásban!")



print("\n4. feladat\n")

szavazatok=list(map(lambda i:jeloltek[i][1],range(len(jeloltek))))
szavdb=sum(szavazatok)
print(f'A választáson {szavdb} állampolgár, a jogosultak {100*szavdb/jogdb:.2f}%-a  vett részt.')



print("\n5. feladat\n")

print("Az egyes pártokra leadott szavazatok aránya:")

rovlista=["GYEP", "HEP", "TISZ", "ZEP","-"]
partlista=["Gyümölcsevők Pártja", "Húsevők Pártja", "Tejivók Szövetsége",
           "Zöldségevők Pártja", "Független jelöltek"]

for i in range(len(rovlista)):
    aktindexek=list(filter(lambda j: jeloltek[j][3]==rovlista[i] ,range(len(jeloltek))))
    aktszavazatok=list(map(lambda j:jeloltek[j][1] ,aktindexek))
    aktszavdb=sum(aktszavazatok)
    print(f'{partlista[i]} = {100*aktszavdb/szavdb:.2f}%')
```

```
print("\n6. feladat\n")

def tamogato(rov):
    if rov=="-":
        return "független"
    else:
        return rov

maxszavjelolt=max(jeloltek,key=lambda jelolt:jelolt[1])
maxszav=maxszavjelolt[1]

print(f'A legtöbb szavazatot ({maxszav}) kapott jelöltek:')

aktjeloltek=list(filter(lambda jelolt: jelolt[1]==maxszav ,jeloltek))
for jelolt in aktjeloltek:
    print(f'{jelolt[2]} {tamogato(jelolt[3])}')



print("\n7. feladat\n")

vkdb=8

kitxt = open("kepviselok.txt","w")

for i in range(1,vkdb+1):
    aktjeloltek=list(filter(lambda jelolt: jelolt[0]==i ,jeloltek))
    kepviselo=max(aktjeloltek,key=lambda jelolt: jelolt[1])
    kitxt.write(f'{i} {kepviselo[2]} {tamogato(kepviselo[3])}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2013maj/EmeltInfo2013maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A helyhatósági választáson 40 képviselőjelölt indult.

3. feladat

Kérem egy képviselő nevét. A vezetékneve: Szilva
Az utóneve: Szilvia
Szilva Szilvia 87 szavazatot kapott.

4. feladat

A választáson 4713 állampolgár, a jogosultak 38.18%-a  vett részt.

5. feladat

Az egyes pártokra leadott szavazatok aránya:
Gyümölcsevők Pártja = 16.36%
Húsevők Pártja = 24.59%
Tejivók Szövetsége = 21.49%
Zöldségevők Pártja = 20.03%
Független jelöltek = 17.53%

6. feladat

A legtöbb szavazatot (288) kapott jelöltek:
Joghurt Jakab TISZ
Narancs Edmond GYEP
Vadas Marcell HEP

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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.
```

```
"""
2013. május idegennyelvű: Számok
@author Klemand66
"""

from random import *


print("\n1. feladat\n")


betxt = open("felszam.txt")

feladatok=[]

while (sor:=betxt.readline())!="":
    feladat=[sor.strip()] # 0: kérdés
    sor=betxt.readline()
    darsor=sor.strip().split()
    feladat+=[int(darsor[0]),int(darsor[1]),darsor[2]]
    # 1: válasz 2: pontszám 3: téma
    feladatok.append(feladat)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'Az adatfájlban {len(feladatok)} feladat van.')



print("\n3. feladat\n")

matfeladatok=list(filter(lambda fel: fel[3]=="matematika", feladatok))

matdb=len(matfeladatok)
mpdb=[0,0,0]

for fel in matfeladatok:
    aktindex=fel[2]-1
    mpdb[aktindex]+=1

print(f'Az adatfájlban {matdb} matematika feladat van, \n\
1 pontot ér {mpdb[0]} feladat, 2 pontot ér {mpdb[1]} feladat, 3 pontot ér {mpdb[2]} feladat.')



print("\n4. feladat\n")

minvfel=min(feladatok, key=lambda fel: fel[1])
maxvfel=max(feladatok, key=lambda fel: fel[1])

print(f'A fájlban a válaszok legkisebb számértéke {minvfel[1]}, a legnagyobb pedig {maxvfel[1]}.')



print("\n5. feladat\n")

temakorok,tkdbk=[],[]

for fel in feladatok:
    if fel[3]  not in temakorok:
        temakorok.append(fel[3])
        tkdbk.append(1)

print(f'Az adatfájlban szereplő témakörök:\n{", ".join(temakorok)}.')
```

```
print("\n6. feladat\n")

akttkor=input(f'Kérem kisbetűvel, ékezetmentesen egy témakör nevét a\n\
{", ".join(temakorok)} témakörök közül: ')

aktfeladatok=list(filter(lambda fel: fel[3]==akttkor, feladatok))

velind=randrange(len(aktfeladatok))
valasz=int(input(f'{aktfeladatok[velind][0]} '))
if valasz==aktfeladatok[velind][1]:
    print(f'A válasz {aktfeladatok[velind][2]} pontot ér.')
else:
    print(f'A válasz 0 pontot ér.\nA helyes válasz: {aktfeladatok[velind][1]}')



print("\n7. feladat\n")

velindexek=[]
while len(velindexek)<10:
    velind=randrange(len(feladatok))
    if velind not in velindexek:
        velindexek.append(velind)

osszpont=0

kitxt = open("tesztfel.txt","w")

for i in velindexek:
    kitxt.write(f'{feladatok[i][2]} {feladatok[i][1]} {feladatok[i][0]}\n')
    osszpont+=feladatok[i][2]

kitxt.write(f'A feladatsorra összesen {osszpont} pont adható.\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2013mid/EmeltInfo2013mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az adatfájlban 49 feladat van.

3. feladat

Az adatfájlban 16 matematika feladat van,
1 pontot ér 5 feladat, 2 pontot ér 7 feladat, 3 pontot ér 4 feladat.

4. feladat

A fájlban a válaszok legkisebb számértéke 2, a legnagyobb pedig 93030.

5. feladat

Az adatfájlban szereplő témakörök:
tortenelem, foldrajz, magyar, kemia, matematika.

6. feladat

Kérem kisbetűvel, ékezetmentesen egy témakör nevét a
tortenelem, foldrajz, magyar, kemia, matematika témakörök közül: kemia
Mennyi egy semleges kemhatasu oldat pH-ja? 7
A válasz 1 pontot ér.

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!



A tesztfel.txt szövegfájl

2 1606 Mikor kotottek meg a zsitvatoroki beket?
2 5050 Mennyi az elso 100 szam osszege?
1 12 Mennyi 4 es 6 legkisebb kozos tobbszorose?
2 1949 Melyik evben alakult meg a NATO?
2 120 Mennyi 5 faktorialisa?
3 77 Hany km hosszu a Balaton?
3 1415 Melyik evben egettek meg Husz Janost?
3 3 Mennyi 1000 tizes alapu logaritmusa?
1 2004 Mikor lett Magyarorszag az Europai Unio tagja?
3 157 Kb. hany ezer negyzetkilometer a Tisza vizgyujto terulete?
A feladatsorra összesen 22 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!

```
"""
2013. október: Közúti ellenőrzés
@author Klemand66
"""

print("\n1. feladat\n")

jogdb=12345

jarmuvek=[]
betxt = open("jarmu.txt")

for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[int(darsor[0]),int(darsor[1]),int(darsor[2]),darsor[3]]
           # 0: óra  1: perc  2: mp  3: rendszám
    jarmuvek.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'Az ellenőrzést végzők legalább {jarmuvek[-1][0]-jarmuvek[0][0]+1} óra hosszat dolgoztak.')


print("\n3. feladat\n")

print("Műszaki ellenőrzések:")
aktora=None
for jarmu in jarmuvek:
    if jarmu[0]!=aktora:
        aktora=jarmu[0]
        print(f'{aktora:2} óra: {jarmu[3]}')



print("\n4. feladat\n")

print("Kategóriánkénti statisztika az ellenőrzőpont előtti áthaladásról:")

kbetuk=["B","K","M"]
tipusok=["autóbusz","kamion","motor"]
szgkdb=len(jarmuvek)

for i in range(len(kbetuk)):
    aktjarmuvek=list(filter(lambda jarmu: jarmu[3][0]==kbetuk[i] ,jarmuvek))
    aktdb=len(aktjarmuvek)
    szgkdb-=aktdb
    print(f'{tipusok[i]}: {aktdb}')

print(f'személygépkocsi: {szgkdb}')



print("\n5. feladat\n")

idok=[]
for jarmu in jarmuvek:
    idok.append(jarmu[0]*3600+jarmu[1]*60+jarmu[2])

print("A leghosszabb forgalommentes időszak az első és az utolsó jármű elhaladása között: ")

maxindex=max(range(1,len(idok)),key=lambda i: idok[i]-idok[i-1])

indexek=[maxindex-1,maxindex]
formidok=[]
for i in indexek:
    formidok.append(f'{jarmuvek[i][0]}:{jarmuvek[i][1]}:{jarmuvek[i][2]}')

print(f'{formidok[0]} - {formidok[1]}')
```

```
print("\n6. feladat\n")

def reszlet(rsz):
    hasznos=""
    for i in joindexek:
        hasznos+=rsz[i]
    return hasznos

hrsz=input("Kérem a rendszámot (7 karakter), a hiányzó karaktereket *-gal jelölve: ")
joindexek=list(filter(lambda i: hrsz[i]!="*" ,range(len(hrsz))))
joreszlet=reszlet(hrsz)

illjarmuvek=list(filter(lambda jarmu: reszlet(jarmu[3])==joreszlet,jarmuvek))

print("Az illeszkedő rendszámok:")

for jarmu in illjarmuvek:
    print(jarmu[3])



print("\n7. feladat\n")

def konvertalo(t):
    mp=t%60
    t=t//60
    perc=t%60
    ora=t//60
    return f'{ora:02} {perc:02} {mp:02}'  # bevezető nullák


kitxt = open("vizsgalt.txt","w")

vizsgkezd=-400
for i,jarmu in enumerate(jarmuvek):
    if idok[i]-vizsgkezd>=300: # az első járműnél biztosan teljesül
        kitxt.write(f'{konvertalo(idok[i])} {jarmu[3]}\n')
        vizsgkezd=idok[i]

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2013okt/EmeltInfo2013okt.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az ellenőrzést végzők legalább 6 óra hosszat dolgoztak.

3. feladat

Műszaki ellenőrzések:
 8 óra: FD-2717
 9 óra: GK-3407
10 óra: RQ-8890
11 óra: IN-5066
12 óra: GC-0459
13 óra: CH-1893

4. feladat

Kategóriánkénti statisztika az ellenőrzőpont előtti áthaladásról:
autóbusz: 10
kamion: 12
motor: 15
személygépkocsi: 317

5. feladat

A leghosszabb forgalommentes időszak az első és az utolsó jármű elhaladása között:
8:57:48 - 9:1:6

6. feladat

Kérem a rendszámot (7 karakter), a hiányzó karaktereket *-gal jelölve: *A****6
Az illeszkedő rendszámok:
YA-7536
QA-4576
NA-2446
QA-4856

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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
```

```
"""
2014. május: IPv6
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("ip.txt")

cimek=[]
for sor in betxt:
    cimek.append(sor.strip())

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'Az állományban {len(cimek)} darab IPv6 cím van.')



print("\n3. feladat\n")

print(f'A legalacsonyabb tárolt IP-cím: {min(cimek)}')



print("\n4. feladat\n")

dok=list(filter(lambda cim: cim[:9]=="2001:0db8",cimek))
glob=list(filter(lambda cim: cim[:7]=="2001:0e",cimek))
helyi=list(filter(lambda cim: cim[:2]=="fc" or cim[:2]=="fd",cimek))

print(f'Dokumentációs cím: {len(dok)} darab')
print(f'Globális egyedi cím: {len(glob)} darab')
print(f'Helyi egyedi cím: {len(helyi)} darab')



print("\n5. feladat\n")


kitxt = open("sok.txt","w")

for i,cim in enumerate(cimek):
    if cim.count("0")>=18:
        kitxt.write(f'{i+1} {cim}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n6. feladat\n")

ssz=int(input("Kérek egy sorszámot (1<= sorszám <= {len(cimek)} "))
cim=cimek[ssz-1]
print(cim)
csoportok=cim.split(":")

for i,csoport in enumerate(csoportok):
    csoport=csoport.lstrip("0")
    # lstrip("k") csak bal oldalról távolítja el a k karaktert, amíg mást nem talál
    if csoport=="": # ha mindegyik 0 volt, egyet vissza kell tenni
        csoport="0"
    csoportok[i]=csoport

print(":".join(csoportok))
```

```
print("\n7. feladat\n")

aktkezdet, akthossz, maxkezdet, maxhossz = None, 0, None, 0

for i,csoport in enumerate(csoportok):
    if csoport=="0":
        if aktkezdet==None:
            aktkezdet=i
        akthossz+=1
        if akthossz> maxhossz:
            maxkezdet=aktkezdet
            maxhossz=akthossz
    else:
        aktkezdet=None
        akthossz=0

if maxhossz>1:
    ujcim=":".join(csoportok[:maxkezdet])+ "::" +":".join(csoportok[maxkezdet+maxhossz:])
    print(ujcim)
else:
    print("Nem rövidíthető tovább.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2014maj/EmeltInfo2014maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az állományban 375 darab IPv6 cím 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: 149 darab

5. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

6. feladat

Kérek egy sorszámot (1<= sorszám <= {len(cimek)} 373
fc11:0000:0000:0f00:0000:0000:0000:2222
fc11:0:0:f00:0:0:0:2222

7. feladat

fc11:0:0:f00::2222

A befejezéshez nyomd meg az ENTER billentyűt!


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
...
```

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
```

```
"""
2014. május idegennyelvű: Céllövészet
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("verseny.txt")

l=int(betxt.readline().strip())
loadatok=[]

for i in range(l):
    sor=betxt.readline().strip()
    loadatok.append(sor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print("Az egymást követően többször találó versenyzők: ")

for i,losor in enumerate(loadatok):
    if "++" in losor:
        print(i+1,end=" ")
print()



print("\n3. feladat\n")

lovesek=list(map(lambda losor:len(losor) ,loadatok))

print(f'A legtöbb lövést leadó versenyző rajtszáma: {lovesek.index(max(lovesek))+1}')



print("\n4. feladat\n")

def loertek(sor):
    aktpont=20
    ertek=0
    for i in range(len(sor)):
        if aktpont>0 and sor[i]=="-":
            aktpont-=1
        else:
            ertek+=aktpont
    return ertek

print(f'Az elkészített függvény:\n\n\
def loertek(sor):\n\
    aktpont=20\n\
    ertek=0\n\
    for i in range(len(sor)):\n\
        if aktpont>0 and sor[i]=="-":\n\
            aktpont-=1\n\
        else:\n\
            ertek+=aktpont\n\
    return ertek')
```

```
print("\n5. feladat\n")

rsz=int(input(f'Adjon meg egy rajtszámot! (1<= rajtszám <= {l}): '))

losor=loadatok[rsz-1]
losorlist=list(losor)
talindexek=list(filter(lambda i:losorlist[i]=="+" ,range(len(losorlist))))
talsszok=list(map(lambda i:str(i+1), range(len(talindexek))))
print(f'5a. feladat: Célt érő lövések: {" ".join(talsszok)}')
print(f'5b. feladat: Az eltalált korongok száma: {len(talindexek)}')

hibatlanok=losor.split("-")
leghosszabb=max(hibatlanok)
"""
ASCII-kód alapján történik az összehasonlítás.
Mivel pl. +++>++, csupa + jel esetén éppen a leghosszabb sorozatot adja meg.
(Érdekesség: ->+, így a max(loadatok) a legrosszabbul kezdődő lövéssorozatot adná meg.)
"""
print(f'5c. feladat: A leghosszabb hibátlan sorozat hossza: {len(leghosszabb)}')

print(f'5d. feladat: A versenyző pontszáma: {loertek(losor)}')



print("\n6. feladat\n")

eredmenyek=[]
for i,losor in enumerate(loadatok):
    eredmenyek.append([i+1,loertek(losor)]) # 0: rajtszám, 1: lőérték

eredmenyek.sort(key=lambda er: er[1], reverse=True)

kitxt = open("sorrend.txt","w")

hely, aktpont=0, eredmenyek[0][1]+1 # mindkét érték rögtön változni fog

for i,er in enumerate(eredmenyek):
    if er[1]<aktpont:
        hely=i+1
        aktpont=er[1]
    kitxt.write(f'{hely:2}\t{er[0]:2}\t{er[1]}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2014mid/EmeltInfo2014mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az egymást követően többször találó versenyzők:
2 4 5 6 7 8 10 12 14 15 16 18 19 20 21

3. feladat

A legtöbb lövést leadó versenyző rajtszáma: 20

4. feladat

Az elkészített függvény:

def loertek(sor):
    aktpont=20
    ertek=0
    for i in range(len(sor)):
        if aktpont>0 and sor[i]=="-":
            aktpont-=1
        else:
            ertek+=aktpont
    return ertek

5. feladat

Adjon meg egy rajtszámot! (1<= rajtszám <= 21): 8
5a. feladat: Célt érő lövések: 1 2 3 4 5
5b. feladat: Az eltalált korongok száma: 5
5c. feladat: A leghosszabb hibátlan sorozat hossza: 3
5d. feladat: A versenyző pontszáma: 90

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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:

```
33221111111111122233
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
…
```

```
"""
2014. október: Nézőtér
@author Klemand66
"""

print("\n1. feladat\n")

"""
Mindkét fájlt sorfolytonosan tároljuk el.
A tarnszformációk:
sor=index//20 +1, szek=index%20 +1
index=(sor-1)*20+(szek-1)
"""

betxt = open("foglaltsag.txt")

foglaltsagok=[]
for sor in betxt:
    foglaltsagok+=list(sor.strip())
betxt.close()

betxt = open("kategoria.txt")
kategoriak=[]
for sor in betxt:
    kategoriak+=list(sor.strip())

betxt.close()
print("A beolvasások és a szövegfájlok lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

sor=int(input("Kérem egy sor számát (max. 15): "))
szek=int(input("Kérem egy szék számát a sorban (max. 20): "))

if foglaltsagok[(sor-1)*20+(szek-1)]=="o":
    foglaltsag="szabad"
else:
    foglaltsag="foglalt"

print(f'A megadott hely {foglaltsag}.')



print("\n3. feladat\n")

edb=foglaltsagok.count("x")

print(f'Az előadásra eddig {edb} jegyet adtak el, ez a nézőtér {100*edb/len(foglaltsagok):.0f}%-a.')



print("\n4. feladat\n")

katlista=["1","2","3","4","5"]
eladottak=[]

for kat in katlista:
    eladott=list(filter(lambda i: kategoriak[i]==kat and foglaltsagok[i]=="x"  ,range(len(foglaltsagok))))
    eladottak.append(len(eladott))

print(f'A legtöbb jegyet a(z) {eladottak.index(max(eladottak))+1}. árkategóriában értékesítették.')



print("\n5. feladat\n")

arlista=[5000,4000,3000,2000,1500]

bevetelek=list(map(lambda i: eladottak[i]*arlista[i], range(len(arlista))))

print(f'A pillanatnyi teljes bevétel {sum(bevetelek):,} Ft.')
```

```
print("\n6. feladat\n")

def egyedulallo(hely):
    szek=hely%20 +1
    if szek==1:
        return foglaltsagok[hely]=="o" and foglaltsagok[hely+1]=="x"
    elif szek==20:
        return foglaltsagok[hely]=="o" and foglaltsagok[hely-1]=="x"
    else:
        return foglaltsagok[hely]=="o" and foglaltsagok[hely-1]=="x" and foglaltsagok[hely+1]=="x"

egyedulallok=list(map(lambda i: egyedulallo(i), range(len(foglaltsagok))))

print(f'A nézőtéren {egyedulallok.count(True)} egyedülálló üres hely van.')



print("\n7. feladat\n")

def valaszto(i):
    if foglaltsagok[i]=="x":
        return "x"
    else:
        return kategoriak[i]

ujfoglaltsagok=list(map(lambda i: valaszto(i) , range(len(foglaltsagok))))

kitxt = open("szabad.txt","w")

for i,ertek in enumerate(ujfoglaltsagok):
    kitxt.write(ertek)
    if (i+1)%20==0:
        kitxt.write("\n")

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2014okt\EmeltInfo2014okt.py ==================

1. feladat

A beolvasások és a szövegfájlok lezárása sikeresen befejeződött.

2. feladat

Kérem egy sor számát (max. 15): 12
Kérem egy szék számát a sorban (max. 20): 8
A megadott hely szabad.

3. feladat

Az előadásra eddig 187 jegyet adtak el, ez a nézőtér 62%-a.

4. feladat

A legtöbb jegyet a(z) 2. árkategóriában értékesítették.

5. feladat

A pillanatnyi teljes bevétel 593,500 Ft.

6. feladat

A nézőtéren 35 egyedülálló üres hely van.

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A szabad.txt szövegfájl

```
xx2x2x1x1x1x11xxxx2x
xxxxxxxxxxxxxxx222x
x2x2x1x1x1x1x1x2x2x2
22x2x1x1x1x1x1x22x2x
xxxxxxxxxxxxxxx2333
xxxxxxxxxxxxxxxxxxxx
33333222211222223333
xxxxxxxxxx22xxxxxxxx
x44xxxxxxxxx3xxxxx44
xxxxxx3xxxxxxx3xx444
x54444433x3333444xxx
555555x4x4x4x4x5x5x5
5xxxxxxxxxxxxxxxx55
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
```

```
"""
2015. május: Expedíció
@author Klemand66
"""


print("\n1. feladat\n")

betxt = open("veetel.txt")

vetelek=[]

for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[int(darsor[0]),int(darsor[1])]
    # 0: nap  1: rádiós
    sor=next(betxt)
    adatsor.append(sor.strip()) # 3: feljegyzés
    vetelek.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'Az első üzenetet rögzítő rádiós: {vetelek[0][1]}, az utolsóé: {vetelek[-1][1]},')



print("\n3. feladat\n")

farkasok=list(filter(lambda i: "farkas" in vetelek[i][2] ,range(len(vetelek))))

for f in farkasok:
    print(f'{vetelek[f][0]}. nap {vetelek[f][1]}. rádióamatőr')



print("\n4. feladat\n")

napdb=11

for n in range(1,napdb+1):
    radiosok=list(filter(lambda i: vetelek[i][0]==n ,range(len(vetelek))))
    print(f'{n:2}. nap: {len(radiosok):2} rádióamatőr')



print("\n5. feladat\n")

kardb=90

def helyreallito(klista):
    for k in klista:
        if k!="#":
            return k
    return "#"

kitxt = open("adaas.txt","w")

for n in range(1,napdb+1):
    napivetelek=list(filter(lambda vetel: vetel[0]==n ,vetelek))
    fj=""
    for i in range(kardb):
        karakterlista=list(map(lambda vetel: vetel[2][i],napivetelek))
        fj+=helyreallito(karakterlista)
    kitxt.write(f'{fj}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")
```

```
print("\n6. feladat\n")

def szame(szo):
    valasz=True
    for i in range(len(szo)):
        if szo[i]<"0" or szo[i]>"9":
            valasz=False
    return valasz

print(f'Az elkészített függvény:\n\n\
def szame(szo):\n\
    valasz=True\n\
    for i in range(len(szo)):\n\
        if szo[i]<"0" or szo[i]>"9":\n\
            valasz=False\n\
    return valasz')



print("\n7. feladat\n")

radiosdb=20
nap=int(input(f'Adja meg a nap sorszámát (1<= nap <= {napdb})! '))
radios=int(input(f'Adja meg a rádióamatőr sorszámát (1<= rádiós <= {radiosdb})! '))
aktvetel=list(filter(lambda vetel: vetel[0]==nap and vetel[1]==radios ,vetelek))

if len(aktvetel)>0:
    aktfj=aktvetel[0][2]
    inf=aktfj.split(" ")[0] # csak a daraboltsor első eleme hordoz információt
    egyedek=inf.split("/")
    if len(egyedek)==2 and egyedek[0].isnumeric() and egyedek[1].isnumeric:
        print(f'A megfigyelt egyedek száma: {int(egyedek[0])+int(egyedek[1])}.')
    else:
        print("Nincs információ.")
else:
    print("Nincs ilyen feljegyzés.")

"""
Mivel a feljegyzésekben a / előtt biztosan van egy karakter,
és utána sem következhet közvetlenül szóköz, nem okozhat hibát,
hogy a szame függvény az üres szót is számnak tekinti.
Pl. az inf="8/" esetén lépne fel futási hiba.
A beépített .isnumeric() metódus "" esetén False értéket ad,
ezért üres karakter esetén sem futnánk bele az int("") hibába.
"""

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2015maj/EmeltInfo2015maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az első üzenetet rögzítő rádiós: 13, az utolsóé: 18,

3. feladat

10. nap 16. rádióamatőr
5. nap 15. rádióamatőr

4. feladat

 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

5. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

6. feladat

Az elkészített függvény:

def szame(szo):
    valasz=True
    for i in range(len(szo)):
        if szo[i]<"0" or szo[i]>"9":
            valasz=False
    return valasz

7. feladat

Adja meg a nap sorszámát (1<= nap <= 11)! 2
Adja meg a rádióamatőr sorszámát (1<= rádiós <= 20)! 15
A megfigyelt egyedek száma: 1.

A befejezéshez nyomd meg az ENTER billentyűt!


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 választják 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!

```
"""
2015. május idegennyelvű: Latin táncok
@author Klemand66
"""

print("\n1. feladat\n")

tancok, lanyok, fiuk=[], [], []

betxt = open("tancrend.txt")

for sor in betxt:
    tancok.append(sor.strip())
    sor=next(betxt)
    lanyok.append(sor.strip())
    sor=next(betxt)
    fiuk.append(sor.strip())

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'Az elsőként bemutatott tánc neve: {tancok[0]}')
print(f'Az utolsóként bemutatott tánc neve: {tancok[-1]}')



print("\n3. feladat\n")

print(f'A sambát {tancok.count("samba")} pár mutatta be.')



print("\n4. feladat\n")

Vilmaindexek=list(filter(lambda i: lanyok[i]=="Vilma" ,range(len(tancok))))
Vilmatancok=list(map(lambda i: tancok[i] ,Vilmaindexek))
print(f'Vilma a következő táncokban szerepelt: {", ".join(Vilmatancok)}.')



print("\n5. feladat\n")

akttanc=input("Kérek egy táncot a cha-cha, salsa, rumba, samba, jive, tango, bachata közül: ")
if akttanc in Vilmatancok:
    for i,fiu in enumerate(fiuk):
        if lanyok[i]=="Vilma" and tancok[i]==akttanc:
            Vilmapartner=fiuk[i]
            break
    print(f'A {akttanc} bemutatóján Vilma párja {Vilmapartner} volt.')
else:
    print(f'Vilma nem táncolt {akttanc}-t.')



print("\n6. feladat\n")

lanylista, fiulista=[], []

for lany in lanyok:
    if lany not in lanylista:
        lanylista.append(lany)

for fiu in fiuk:
    if fiu not in fiulista:
        fiulista.append(fiu)

kitxt = open("szereplok.txt","w")

kitxt.write(f'Lányok: {", ".join(lanylista)}\n')
kitxt.write(f'Fiúk: {", ".join(fiulista)}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")
```

```
print("\n7. feladat\n")

print("A legtöbbet szereplők:")

lanydbk=list(map(lambda lany:lanyok.count(lany) ,lanylista))
maxlanydb=max(lanydbk)
maxlanyok=[]

for i,lany in enumerate(lanylista):
    if lanydbk[i]==maxlanydb:
        maxlanyok.append(lany)

print(f'Lányok: {", ".join(maxlanyok)} ({maxlanydb} táccal)')

fiudbk=list(map(lambda fiu:fiuk.count(fiu) ,fiulista))
maxfiudb=max(fiudbk)
maxfiuk=[]

for i,fiu in enumerate(fiulista):
    if fiudbk[i]==maxfiudb:
        maxfiuk.append(fiu)

print(f'Fiúk: {", ".join(maxfiuk)} ({maxfiudb} táccal)')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2015mid/EmeltInfo2015mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az elsőként bemutatott tánc neve: cha-cha
Az utolsóként bemutatott tánc neve: bachata

3. feladat

A sambát 10 pár mutatta be.

4. feladat

Vilma a következő táncokban szerepelt: cha-cha, rumba, tango.

5. feladat

Kérek egy táncot a cha-cha, salsa, rumba, samba, jive, tango, bachata közül: tango
A tango bemutatóján Vilma párja Tivadar volt.

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

7. feladat

A legtöbbet szereplők:
Lányok: Katalin, Szilvia (7 táccal)
Fiúk: Kelemen, Bertalan (7 táccal)

A befejezéshez nyomd meg az ENTER billentyűt!
```

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.
```

```
"""
2015. október: Fej vagy írás
@author Klemand66
"""

from random import *


print("\n1. feladat\n")

erme=["I","F"]
dobas=choice(erme) # véletlen választás a lista elemei közül

print(f'A pénzfeldobás eredménye: {dobas}')



print("\n2. feladat\n")

tipp=input("Tippeljen! (F/I)= ").strip().upper() # a kisbetűs beírást is feldolgozzuk
dobas=choice(erme)
print(f'A tipp {tipp}, a dobás eredménye {dobas} volt.')
if tipp==dobas:
    print("Ön eltalálta.")
else:
    print("Ön nem találta el.")



print("\n3. feladat\n")

betxt = open("kiserlet.txt")

db,fejdb=0,0

for sor in betxt:
    db+=1
    if sor.strip()=="F": # a fájlban csak nagybetűk vannak
       fejdb+=1

betxt.close()

print(f'A kísérlet {db} dobásból állt.')



print("\n4. feladat\n")

print(f'A kísérlet során a fej relatív gyakorisága {100*fejdb/db:.2f}% volt.')



print("\n5. feladat\n")

betxt = open("kiserlet.txt")

db2, aktsor=0, "I" # kezdőérték (az FFI-vel kezdődő fájlok feldolgozásához kell)

for sor in betxt:
    sor=sor.strip()
    if sor=="I":
        if aktsor=="IFF":
            db2+=1
        aktsor="I"
    else:
        aktsor+="F"

betxt.close()

print(f'A kísérlet során {db2} alkalommal dobtak pontosan két fejet egymás után.')
```

```
print("\n6. feladat\n")

betxt = open("kiserlet.txt")

aktkezdet, akthossz, maxkezdet, maxhossz = None, 0, None, 0

i=0
for sor in betxt:
    i+=1
    sor=sor.strip()
    if sor=="F":
        if aktkezdet==None:
            aktkezdet=i
        akthossz+=1
        if akthossz> maxhossz:
            maxkezdet=aktkezdet
            maxhossz=akthossz
    else:
        aktkezdet=None
        akthossz=0

print(f'A leghosszabb tisztafej sorozat {maxhossz} tagból áll, kezdete a(z) {maxkezdet}. dobás.')

betxt.close()



print("\n7. feladat\n")

dbf, dbi, sorozat=0, 0, []

for i in range(1000):
    dobasok=""
    for j in range(4):
        dobasok+=choice(erme)
    if dobasok=="FFFF":
        dbf+=1
    if dobasok=="FFFI":
        dbi+=1
    sorozat.append(dobasok)

kitxt = open("dobasok.txt","w")

kitxt.write(f'FFFF: {dbf}, FFFI: {dbi}\n')
kitxt.write(f'{" ".join(sorozat)}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2015okt\EmeltInfo2015okt.py ==================

1. feladat

A pénzfeldobás eredménye: F

2. feladat

Tippeljen! (F/I)= f
A tipp F, a dobás eredménye F 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.

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!


A dobasok.txt szövegfájl

FFFF: 77, FFFI: 74
FFII IIIF IFII IFFF FIFF FIFI IIFI FIFI FIFF IIFF IFFF FFII IIIF IIFI IFFF FIFI FFFF FFFF IFIF IFFF IIFI FIFI
FFFI IFFF FFFF IFII FFIF IIIF IFFI IIFI IFFF IIIF IIIF FIIF IIIF IFFF IFIF FIII FIII FIFI FFFI IIFF IFFI IIIF
FFII IFII IIII IFFF FIFF IIFI FFFF IFII FFII IFIF FIII IFFI FFFI IFFF FIIF IFIF FIFF IFFI FFFF FFFI IFFF IFIF
IFFI FFFF FIII IIFI IFIF FIFI FFIF FIFF IIFI FFFI FFFF IIII IFII FFII FIFF FIFI IFIF FIFI FIIF IFIF FFFF IFIF
IIFI FFFI FIFF FFFF FFIF FFFI IFFF IIFI IFIF IIFI IFII IIII FIFF IFFF IIFF FIII FFII FIFI FIII IFFF IFII IIFI
FIIF IFIF IFFF IIII IIFF FIFI IFII IIIF FFFF IIIF IFIF IFII IIIF FFFI FIII FIII FIFF FFII FIFF IIFF FFFF FFIF
FFIF IFII FIII IIFI IFII IFFI IFIF IIIF IFIF IIFI IIII FFFF IFIF IFFI FFFF IFFI IIIF IFFF FIII IIFI IIII FFIF
FIII FIFF IFFF FIFI FIIF IFFF FFFI IFFF FIIF FFIF IFFI FIFF FIFI FFFF FFII FIIF FFFI IIFF FFIF IIFI FIFI FFIF
FFFF FIFI FFFF FIFI IIIF IFFI IIFF IFFF FFFF FIII FIFF IIII FIIF FIII IFIF FFFI FFII IFFI FFIF IIIF FIIF IFFF
FFFF FFFI FIFF IIII FFIF FIFI IIIF IFII FIII IIFF FFFF FFFF IIFI FFFI FFFF FIFI IFFF IIII FIFF FIII IIIF IIFF
FFII IIII FIII FFFI FIFF IIFF FFFI IFFF FIIF FFIF IFIF FIII IIFF IIFI FIFF FIFF FFIF FFIF FIFF IIIF FFFF FFIF
IIIF FIFI IIFI FFII IFFI IIFF IFFF IFIF FIFF IFIF IFIF FIFI IFIF IFII IFIF FFIF IIFF IIIF IFFF FIIF IFII IIII
FFII IIII FFIF FFFI FFII IFIF FIFI FIII IFIF IIFI FFIF FIII FIFF FFIF IFFI IIFF IFII IFFF FIII IFII IIII IFFI
FIFF FFFF FFII IIIF FFII FIFI FIFF IFII FIII IIFI FIFF FFIF IFFF IFIF IIIF IIFI IIII FFIF FIFF IFFF FFII IIFI
FIFI FFFI IIIF FIFF FFII FIIF FFFF IFFI IIII FFII IIFF FFFF FFII FFIF FFFI FFFF IFII FIIF IFII FFIF FFIF FIIF
FIIF FIIF IIFI IFFF IIFI IIIF FFFF IFII IIFF FIII FFII IIFF FIFF IIFI FIFF IIFI IIFI FFIF IIFF IIFI FFII IIII
FIIF IFFI IFFI FFFI FIIF IIII FFII FFFI FIFF FFFI FIIF FFIF FIIF FFIF FFFF FFFF FIFF IFFI FFIF FIFF FFFI IIFF
IFFF IIFF FIFF FFFF IFIF IFFF FFII FFIF IIIF FIFF IIIF IFII IIFI IFFF FIFI FFFI FIFF IFFF IIII FIFI IFII IIFF
FFFF IFIF IIII FIIF FFFF FFFF FFIF IIFF IFII FFFI IFII IIFI FIII IFFF IIII FFIF FIIF IIFF FFIF IIFF FIFF IFII
IFIF IIFI IFII FIIF IFII FIII IFFF FFFF IIII IFIF FFFI IFFF IFIF FFIF FFFI FIII FIFF IFII FFFF IIFI FIIF IIFF
IIFF FFFI FFFI IFFF IIFF FFIF IFII FFFI FIIF IFIF IFII IIFI FFFI IFFI IIFF IIII FIII FIII IIFI IFFI FFII FIIF
IFIF IIIF IIFF FFII FFFF IFFI IFIF IFFF IFFI FFFF IFII FFFF IIFI IIII IFII IFIF IFFF IFIF IIFF IFFF IIFI IIFF
FFFF FIII IFII IIFI IIFF IIII IIFI FIFF IFFF IIFF IIFF IFFI IFFI IFFI FFFF FFFI IFFF FFFF IIFF IFFI IIII IIII
IFIF FFFI IIFF IIIF FIFI IFII IIFI IFIF FFIF FIIF IFFF FFFI FIFI FIFF IFIF FIFI IFFI IFIF IIFI FFII IIII FIIF
FFFI FFFF IIFI IFII IFIF FIIF FIFF IFIF FIFF IIII IIFF FFFF FIIF IIFF IIII FFII IFFF FFIF FIFF FIIF FIFI FFII
IIIF FFII FFIF FFII IIFI IIFF IIIF IFII IIIF IIII FIFF IIFI FIIF IIIF IIIF FIFI IFFF IIIF FIIF FIII IIFI FFFI
...
```

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
…
```

```
"""
2016. május: Ötszáz
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("penztar.txt")

kosarak, cikkek = [], []

for sor in betxt:
    sor=sor.strip()
    if sor=="F":
        kosarak.append(cikkek)
        cikkek=[]
    else:
        cikkek.append(sor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A fizetések száma: {len(kosarak)}')



print("\n3. feladat\n")

print(f'Az első vásárló {len(kosarak[0])} darab árucikket vásárolt.')



print("\n4. feladat\n")

aktssz=int(input(f'Kérem egy vásárlás sorszámát (1 <= sorszám <= {len(kosarak)}): '))
aktcikk=input(f'Kérem egy árucikk nevét (Pl. toll, colostok, szatyor): ')
aktdb=int(input(f'Kérem a vásárolt darabszámot (1 <= db <= 20): '))



print("\n5. feladat\n")

aktindexek=list(filter(lambda i: aktcikk in kosarak[i],range(len(kosarak))))
print(f'Az első vásárlás sorszáma: {aktindexek[0]+1}')
print(f'Az utolsó vásárlás sorszáma: {aktindexek[-1]+1}')
print(f'{len(aktindexek)} vásárlás során vettek belőle.')



print("\n6. feladat\n")

def ertek(darab):
    if darab==1:
        return 500
    elif darab==2:
        return 950
    else:
        return 950+(darab-2)*400

print(f'{aktdb} darab azonos árucikk vételekor fizetendő összeg: {ertek(aktdb)} Ft.')
```

```
print("\n7. feladat\n")

# Egy konkrét vásárlás feldolgozása:
kosar=kosarak[aktssz-1]
cikklista, dblista= [], []

for i,cikk in enumerate(kosar):
    if cikk in cikklista:
        aktindex=cikklista.index(cikk)
        dblista[aktindex]+=1
    else:
        cikklista.append(cikk)
        dblista.append(1)

for i in range(len(cikklista)):
    print(f'{dblista[i]} {cikklista[i]}')



print("\n8. feladat\n")

# Sz előbbi algoritmus az összes vásárlásra kiterjesztve:
cikklistak, dblistak= [], []

for kosar in kosarak:
    cikklista, dblista= [], []
    for i,cikk in enumerate(kosar):
        if cikk in cikklista:
            aktindex=cikklista.index(cikk)
            dblista[aktindex]+=1
        else:
            cikklista.append(cikk)
            dblista.append(1)
    cikklistak.append(cikklista)
    dblistak.append(dblista)

kitxt = open("osszeg.txt","w")

for i,dblista in enumerate(dblistak):
    arlista=list(map(lambda db: ertek(db),dblista))
    kitxt.write(f'{i+1}: {sum(arlista)}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

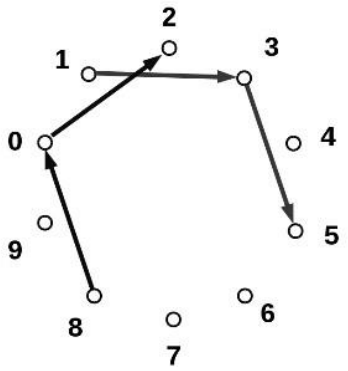
```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2016maj/EmeltInfo2016maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A fizetések száma: 141

3. feladat

Az első vásárló 1 darab árucikket vásárolt.

4. feladat

Kérem egy vásárlás sorszámát (1 <= sorszám <= 141): 2
Kérem egy árucikk nevét (Pl. toll, colostok, szatyor): kefe
Kérem a vásárolt darabszámot (1 <= db <= 20): 2

5. feladat

Az első vásárlás sorszáma: 5
Az utolsó vásárlás sorszáma: 139
32 vásárlás során vettek belőle.

6. feladat

2 darab azonos árucikk vételekor fizetendő összeg: 950 Ft.

7. feladat

2 colostok
2 HB ceruza
1 toll
1 szatyor
1 csavarkulcs
1 doboz

8. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

Az osszeg.txt szövegfájl

1: 500	19: 1000	37: 1000	55: 500
2: 3900	20: 1950	38: 1000	56: 500
3: 2300	21: 500	39: 1500	57: 500
4: 1000	22: 500	40: 500	58: 1000
5: 2500	23: 1000	41: 500	59: 1500
6: 2900	24: 1450	42: 2500	60: 500
7: 950	25: 500	43: 1500	61: 2500
8: 1950	26: 4750	44: 500	62: 500
9: 3750	27: 500	45: 1500	63: 500
10: 1500	28: 1500	46: 2450	64: 1000
11: 500	29: 1450	47: 1500	65: 1000
12: 1950	30: 1000	48: 500	66: 1000
13: 500	31: 500	49: 1000	67: 1000
14: 500	32: 1000	50: 1000	68: 500
15: 1000	33: 1500	51: 500	69: 950
16: 500	34: 1950	52: 1500	70: 2900
17: 500	35: 1000	53: 1000	...
18: 1000	36: 500	54: 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
…
```

```
"""
2016. május idegennyelvű: Zár
@author Klemand66
"""

from random import *


print("\n1. feladat\n")

betxt = open("ajto.txt")

kodok=[]
for sor in betxt:
    kodok.append(sor.strip())

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

nyito=input("Adja meg, mi nyitja a zárat (min. 3, max. 10 karakter, pl. 239451)! ")



print("\n3. feladat\n")

nyitok=list(filter(lambda i: kodok[i]==nyito,range(len(kodok))))
nyitokstr=list(map(lambda i: str(i+1) ,nyitok))
print(f'A nyitó kódszámok sorai: {" ".join(nyitokstr)}')



print("\n4. feladat\n")

def ismetlodo(kod):
    for kar in kod:
        if kod.count(kar)>1:
            return True
    return False

van, ssz = False, None
for i,kod in enumerate(kodok):
    if ismetlodo(kod):
        van=True
        ssz=i+1
        break

if van:
    print(f'Az első ismétlődést tartalmazó próbálkozás sorszáma: {ssz}')
else:
    print("Nem volt ismétlődő számjegy.")



print("\n5. feladat\n")

velkod=""
while len(velkod)<len(nyito):
    szamjegy=randrange(10)
    if str(szamjegy) not in velkod:
        velkod+=str(szamjegy)

print(f'Egy {len(nyito)} hosszúságú kódszám: {velkod}')
```

```
print("\n6. feladat\n")

def nyit(jo, proba):
    egyezik = len(jo)==len(proba)
    if egyezik:
        elteres = ord(jo[0])-ord(proba[0])
        for i in range(1,len(jo)):
            if (elteres - (ord(jo[i]) - ord(proba[i])))%10 != 0:
                egyezik = False
    return egyezik

print(f'Az elkészített függvény:\n\n\
def nyit(jo, proba):\n\
    egyezik = len(jo)==len(proba)\n\
    if egyezik:\n\
        elteres = ord(jo[0])-ord(proba[0])\n\
        for i in range(1,len(jo)):\n\
            if (elteres - (ord(jo[i]) - ord(proba[i]))%10 != 0:\n\
                egyezik = False\n\
     return egyezik')



print("\n7. feladat\n")

kitxt = open("siker.txt","w")

for kod in kodok:
    kitxt.write(f'{kod} ')
    if len(kod)==len(nyito):
        if nyit(nyito,kod):
            kitxt.write(f'sikeres\n')
        else:
            kitxt.write(f'hibás kódszám\n')
    else:
        kitxt.write(f'hibás hossz\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2016mid/EmeltInfo2016mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Adja meg, mi nyitja a zárat (min. 3, max. 10 karakter, pl. 239451)! 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 hosszúságú kódszám: 401236

6. feladat

Az elkészített függvény:

def nyit(jo, proba):
    egyezik = len(jo)==len(proba)
    if egyezik:
        elteres = ord(jo[0])-ord(proba[0])
        for i in range(1,len(jo)):
            if (elteres - (ord(jo[i]) - ord(proba[i]))%10 != 0:
                egyezik = False
     return egyezik

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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.
```

```
"""
2016. október: Telefonos ügyfélszolgálat
@author Klemand66
"""


print("\n1. feladat\n")

def mpbe(o,p,mp):
    return o*3600+p*60+mp

print(f'Az elkészített függvény:\n\n\
def mpbe(o,p,mp):\n\
    return o*3600+p*60+mp')



print("\n2. feladat\n")

betxt = open("hivas.txt")

hivasok=[]

for sor in betxt:
    darsor=sor.strip().split()
    intsor=list(map(int ,darsor))
    adatsor=intsor[:3]+[mpbe(intsor[0],intsor[1],intsor[2])]\
          +intsor[3:]+[mpbe(intsor[3],intsor[4],intsor[5])]
    #0: kó 1: kp 2: kmp 3: kmpbe 4: vó 5: vp 6: vmp 7: vmpbe
    hivasok.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n3. feladat\n")

oradbk=[]
for i in range(24):
    oradbk.append(0)

for hivas in hivasok:
    aktindex=hivas[0]
    oradbk[aktindex]+=1

print("Az óránkénti hívások száma: ")
for i in range(24):
    if oradbk[i]>0:
        print(f'{i:2} óra {oradbk[i]:2} hívás')



print("\n4. feladat\n")

maxind=max(range(len(hivasok)),key=lambda i: hivasok[i][7]-hivasok[i][3])
print(f'A leghosszabb ideig vonalban levő hívó a(z) {maxind+1}. sorban szerepel,\n\
a hívás hossza {hivasok[maxind][7]-hivasok[maxind][3]} másodperc.')
```

```
print("\n5. feladat\n")

idosor=input("Kérek egy időpontot 8 és 12 óra között óra perc mp alakban: ")
darido=idosor.strip().split()
o,p,mp=int(darido[0]),int(darido[1]),int(darido[2])
aktido=mpbe(o,p,mp)

"""
Azok közül, akiknél a híváskezdés ideje <= az aktuális időpontnál, a bontásideje viszont >=,
a legkisebb sorszámú éppen beszél, az utána következők pedig várakoznak.
"""
aktindexek=list(filter(lambda i: hivasok[i][3]<=aktido and  hivasok[i][7]>=aktido ,range(len(hivasok))))
if len(aktindexek)>0:
    besz=aktindexek[0]+1
    varakozok=list(map(lambda i: str(i+1),aktindexek[1:]))
    print(f'A(z) {besz}. hívó beszél és {len(varakozok)} várakozó van.\n\
(A várakozók: {"., ".join(varakozok)}.)')
else:
    print("Nem volt beszélő.")



print("\n6. feladat\n")

"""
Először kiszűrjük azoknak a hívóknak az indexét, akik aktívak voltak munkaidőben,
azaz hívásukat 12 óra előtt kezdték és 8 óra után bontották a vonalat.
Második lépésben pedig azokét, akik az összes előttük lévő hívás bontása után fejezték be a hívásukat.
Ők mind beszéltek, pontosan az indexük sorrendjében.
"""
munkaidoben=list(filter(lambda i: hivasok[i][3]<=12*3600 and hivasok[i][7]>8*3600 ,range(len(hivasok))))
beszelok=list(filter(lambda i:  hivasok[i]==max(hivasok[:i+1],key=lambda hivas: hivas[7]) ,munkaidoben))
varakozas=max(0, hivasok[beszelok[-2]][7]-hivasok[beszelok[-1]][3])

print(f'Az utolsó telefonáló adatai a(z) {beszelok[-1]+1}. sorban vannak, {varakozas} másodpercig várt.')



print("\n7. feladat\n")

kitxt = open("sikeres.txt","w")

def konvertalo(t):
    mp=t%60
    t=t//60
    perc=t%60
    ora=t//60
    return f'{ora:2} {perc:2} {mp:2}'

kezd=konvertalo(8*3600)
for i in beszelok:
    vege=konvertalo(hivasok[i][7])
    kitxt.write(f'{i+1:3} {kezd} {vege}\n')
    kezd=vege

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2016okt/EmeltInfo2016okt.py ==================

1. feladat

Az elkészített függvény:

def mpbe(o,p,mp):
    return o*3600+p*60+mp

2. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

3. feladat

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

4. feladat

A leghosszabb ideig vonalban levő hívó a(z) 152. sorban szerepel,
a hívás hossza 341 másodperc.

5. feladat

Kérek egy időpontot 8 és 12 óra között óra perc mp alakban: 11 59 56
A(z) 425. hívó beszél és 5 várakozó van.
(A várakozók: 428., 429., 430., 432., 433.)

6. feladat

Az utolsó telefonáló adatai a(z) 432. sorban vannak, 184 másodpercig várt.

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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
106  8  6 37  8  6 51
108  8  6 51  8 10 57
112  8 10 57  8 12 43
113  8 12 43  8 13 30
114  8 13 30  8 15 43
115  8 15 43  8 17 13
117  8 17 13  8 17 38
119  8 17 38  8 18  4
120  8 18  4  8 20 39
122  8 20 39  8 20 40
123  8 20 40  8 22 17
126  8 22 17  8 24 51
130  8 24 51  8 25 54
131  8 25 54  8 29 40
137  8 29 40  8 31 42
138  8 31 42  8 33 27
139  8 33 27  8 34 10
143  8 34 10  8 34 26
145  8 34 26  8 35  0
146  8 35  0  8 35 40
148  8 35 40  8 39 22
150  8 39 22  8 41 11
152  8 41 11  8 43 42
154  8 43 42  8 44 23
156  8 44 23  8 45 42
159  8 45 42  8 45 53
160  8 45 53  8 47 47
162  8 47 47  8 49  1
166  8 49  1  8 50 44
167  8 50 44  8 52 29
168  8 52 29  8 55 22
169  8 55 22  8 56  5
173  8 56  5  8 56 53
174  8 56 53  8 57 42
176  8 57 42  9  1 51
178  9  1 51  9  2 27
182  9  2 27  9  6  1
183  9  6  1  9  7 29
184  9  7 29  9  8 42
187  9  8 42  9  8 57
190  9  8 57  9 10  4
191  9 10  4  9 10 10
193  9 10 10  9 11 25
195  9 11 25  9 13 12
198  9 13 12  9 13 38
200  9 13 38  9 14 44
201  9 14 44  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
215  9 26 28  9 28  6
218  9 28  6  9 28 43
220  9 28 43  9 29 24
224  9 29 24  9 33 14
225  9 33 14  9 34 13
227  9 34 13  9 35 55
232  9 35 55  9 38 50
234  9 38 50  9 39 28
235  9 39 28  9 39 44
237  9 39 44  9 41 36
238  9 41 36  9 41 52
239  9 41 52  9 43 46
242  9 43 46  9 48  8
243  9 48  8  9 48 47
244  9 48 47  9 49 46
247  9 49 46  9 50  1
248  9 50  1  9 54  9
250  9 54  9  9 55 39
253  9 55 39  9 58  1
254  9 58  1 10  0 22
255 10  0 22 10  1 15
257 10  1 15 10  4 22
262 10  4 22 10  6 41
268 10  6 41 10  9 30
271 10  9 30 10 10 48
272 10 10 48 10 13 37
277 10 13 37 10 16 18
280 10 16 18 10 19  2
281 10 19  2 10 20 10
282 10 20 10 10 21 18
284 10 21 18 10 22 37
286 10 22 37 10 25 45
290 10 25 45 10 25 47
291 10 25 47 10 26 55
293 10 26 55 10 28 45
300 10 28 45 10 30 20
302 10 30 20 10 31 25
304 10 31 25 10 35 31
308 10 35 31 10 38 18
309 10 38 18 10 40  0
312 10 40  0 10 40 28
314 10 40 28 10 42 59
315 10 42 59 10 45  6
317 10 45  6 10 45 50
318 10 45 50 10 46 17
320 10 46 17 10 48  0
321 10 48  0 10 48  7
323 10 48  7 10 49  4
324 10 49  4 10 51 12
326 10 51 12 10 51 44
327 10 51 44 10 52 15
329 10 52 15 10 54  0
334 10 54  0 10 54 53
335 10 54 53 10 56  1
336 10 56  1 10 56  4
337 10 56  4 10 59 34
341 10 59 34 11  1 46
342 11  1 46 11  2 44
346 11  2 44 11  3 17
347 11  3 17 11  5 29
350 11  5 29 11  5 40
352 11  5 40 11  8 54
354 11  8 54 11  9 50
355 11  9 50 11 12 25
359 11 12 25 11 14 29
360 11 14 29 11 15 15
361 11 15 15 11 15 52
362 11 15 52 11 18 59
367 11 18 59 11 19 37
368 11 19 37 11 20 27
372 11 20 27 11 24 19
374 11 24 19 11 24 48
376 11 24 48 11 25 14
377 11 25 14 11 26 52
378 11 26 52 11 29 18
380 11 29 18 11 30  6
383 11 30  6 11 32 32
384 11 32 32 11 33 36
386 11 33 36 11 36  4
389 11 36  4 11 37  5
391 11 37  5 11 38 57
393 11 38 57 11 41 13
402 11 41 13 11 43 50
404 11 43 50 11 44  0
406 11 44  0 11 45 48
407 11 45 48 11 47 27
409 11 47 27 11 47 51
410 11 47 51 11 48 54
412 11 48 54 11 51 36
413 11 51 36 11 52 18
415 11 52 18 11 53 12
416 11 53 12 11 53 34
417 11 53 34 11 54 26
419 11 54 26 11 55 25
420 11 55 25 11 55 37
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
```

```
"""
2017. május: Tesztverseny
@author Klemand66
"""


print("\n1. feladat\n")



betxt = open("valaszok.txt")

sor=betxt.readline()
megoldas=sor.strip()

kodok, valaszok = [], []
for sor in betxt:
    darsor=sor.strip().split()
    kodok.append(darsor[0])
    valaszok.append(darsor[1])

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A vetélkedőn {len(kodok)} versenyző indult.')



print("\n3. feladat\n")

aktkod=input("Kérem egy versenyző azonosítóját ( Pl. AB123 AH97 JO001): ")

aktindex=kodok.index(aktkod) # feltételezhettük, hogy létezik a megadott azonosító
aktvalasz=valaszok[aktindex]

print(f'{aktvalasz}\t(a versenyző válasza)')



print("\n4. feladat\n")

def ertek(i):
    if aktvalasz[i]==megoldas[i]:
        return "+"
    else:
        return " "

print(f'{megoldas}\t(a helyes megoldás)')
helyesek=list(map(lambda i: ertek(i) ,range(len(aktvalasz))))
print(f'{"".join(helyesek)}\t(a versenyző helyes válaszai)')



print("\n5. feladat\n")

aktssz=int(input(f'Kérem egy feladat sorszámát (1 <= sorszám <= {len(megoldas)} ): '))

jok=list(filter(lambda i: valaszok[i][aktssz-1]==megoldas[aktssz-1] ,range(len(valaszok))))
jodb=len(jok)
print(f'A feladatra {jodb} fő, a versenyzők {100*jodb/len(valaszok):.2f}%-a adott helyes választ.')
```

```
print("\n6. feladat\n")

def sav(i):
    ihat=[0,5,10,13,14] # indexhatárok
    bennevan=list(map(lambda j: ihat[j-1]<=i and i<ihat[j] ,range(1,len(ihat))))
    return bennevan.index(True)

def pontozo(valasz):
    pontlista=[3,4,5,6]
    """
    A pontlista=[3,3,3,3,3,4,4,4,4,4,5,5,5,6] megadás esetén
    a pontoknál közvetlenül pontlista[i]-re hivatkozhatnánk,
    de egy hosszabb lista esetén az már nem lenne elegáns megoldás.
    """
    jok=list(filter(lambda i: valasz[i]==megoldas[i],range(len(valasz))))
    pontok=list(map(lambda i: pontlista[sav(i)] ,jok))
    return sum(pontok)

kitxt = open("pontok.txt","w")

for i in range(len(kodok)):
    kitxt.write(f'{kodok[i]} {pontozo(valaszok[i])}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n7. feladat\n")

eredmenyek=[]
for i in range(len(kodok)):
    eredmenyek.append([kodok[i],pontozo(valaszok[i])])

eredmenyek.sort(key=lambda er: er[1], reverse=True)

dij, aktpont=0, eredmenyek[0][1]+1 # mindkét érték rögtön változni fog

for i,er in enumerate(eredmenyek):
    if er[1]<aktpont:
        dij=dij+1
        aktpont=er[1]
    if dij<=3:
        print(f'{dij}. díj ({aktpont} pont): {er[0]}')
    else:
        break


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2017maj/EmeltInfo2017maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A vetélkedőn 303 versenyző indult.

3. feladat

Kérem egy versenyző azonosítóját ( Pl. AB123 AH97 JO001): AB123
BXCDBBACACADBC  (a versenyző válasza)

4. feladat

BCCCDBBBBCDAAA  (a helyes megoldás)
+ +  +   +      (a versenyző helyes válaszai)

5. feladat

Kérem egy feladat sorszámát (1 <= sorszám <= 14 ): 10
A feladatra 111 fő, a versenyzők 36.63%-a adott helyes választ.

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

7. feladat

1. díj (56 pont): JO001
2. díj (52 pont): DG490
2. díj (52 pont): UA889
3. díj (49 pont): FX387

A befejezéshez nyomd meg az ENTER billentyűt!
```

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
...
```

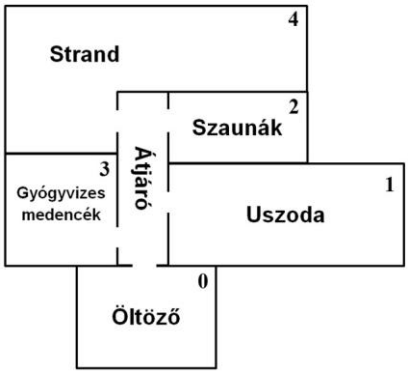
2017. május, idegennyelvű: Fürdő

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
```

```
"""
2017. május idegennyelvű: Fürdő
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("furdoadat.txt")

mozgasok=[]

for sor in betxt:
    darsor=sor.strip().split()
    adatsor=list(map(int ,darsor))
    # 0: az 1: részleg 2: be(0)/ki(1) 3: óra 4:perc 5 mp
    mozgasok.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

oltki=list(filter(lambda mozgas: mozgas[1]==0 and mozgas[2]==1 ,mozgasok))

print(f'Az első vendég {oltki[0][3]}:{oltki[0][4]}:{oltki[0][5]}-kor lépett ki az öltözőből.')
print(f'Az utolsó vendég {oltki[-1][3]}:{oltki[-1][4]}:{oltki[-1][5]}-kor lépett ki az öltözőből.')



print("\n3. feladat\n")

azlista, azdblista = [], []

for m in mozgasok:
    if m[0] in azlista:
        aktindex=azlista.index(m[0])
        azdblista[aktindex]+=1
    else:
        azlista.append(m[0])
        azdblista.append(1)

print(f'A fürdőben {azdblista.count(4)} vendég járt csak egy részlegben.')



print("\n4. feladat\n")

def konvertalo(t):
    mp=t%60
    t=t//60
    perc=t%60
    ora=t//60
    return f'{ora:2}:{perc:2}:{mp:2}'

idolista = []

for az in azlista:
    am=list(filter(lambda m: m[0]==az ,mozgasok))
    kezd=am[0][3]*3600+am[0][4]*60+am[0][5]
    vege=am[-1][3]*3600+am[-1][4]*60+am[-1][5]
    idolista.append(vege - kezd)

maxido=max(idolista)
maxindex=idolista.index(maxido)

print("A legtöbb időt eltöltő vendég:")
print(f'{azlista[maxindex]}. vendég {konvertalo(maxido)}')
```

```
print("\n5. feladat\n")

def sav(erko):
    bennevan=list(map(lambda i: ohat[i-1]<=erko and erko<ohat[i] ,range(1,len(ohat))))
    return bennevan.index(True)

ohat=[6,9,16,20]
dbk=[0,0,0]

for az in azlista:
    am=list(filter(lambda m: m[0]==az ,mozgasok))
    erko=am[0][3]
    dbk[sav(erko)]+=1

for i in range(len(dbk)):
    print(f'{ohat[i]}-{ohat[i+1]} óra között {dbk[i]} vendég')



print("\n6. feladat\n")

kitxt = open("szauna.txt","w")

for az in azlista:
    szm=list(filter(lambda m: m[0]==az and m[1]==2 ,mozgasok))
    szido=0
    for i in range(len(szm)):
        aktido=szm[i][3]*3600+szm[i][4]*60+szm[i][5]
        if i%2==1:
            szido+=aktido
        else:
            szido-=aktido
    if szido>0:
        kitxt.write(f'{az} {konvertalo(szido)}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n7. feladat\n")

rkodok=[0,1,2,3,4]
reszlegek=["Öltöző","Uszoda","Szaunák","Gyógyvizes medencék","Strand"]
rdbk=[0,0,0,0,0]

for az in azlista:
    am=list(filter(lambda m: m[0]==az ,mozgasok))
    rlista=[]
    for i in range(len(am)):
        if am[i][1] not in rlista:
            rlista.append(am[i][1])
    for r in rlista:
        aktindex=rkodok.index(r)
        rdbk[aktindex]+=1

for i in range(1,len(reszlegek)):
    print(f'{reszlegek[i]}: {rdbk[i]}')

print(f'\n(Ellenőrzésképpen: {reszlegek[0]}: {rdbk[0]})')

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2017mid/EmeltInfo2017mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

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 részlegben.

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

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

7. feladat

Uszoda: 41
Szaunák: 52
Gyógyvizes medencék: 54
Strand: 48

(Ellenőrzésképpen: Öltöző: 100)

A befejezéshez nyomd meg az ENTER billentyűt!


A szauna.txt szövegfájl

317  0:14:33                443  0:14:34                287  0:14:41
332  0: 9:55                392  0: 6:25                117  0:11:56
421  0: 8: 4                171  0: 9:31                425  0:14:38
435  0:12:41                147  0: 9:13                257  0:11:15
217  0: 5:32                206  0:22:34                204  0:11: 7
130  0:14:16                191  0:10:27                389  0:11:44
459  0:13:13                210  0:10:31                451  0:24:17
480  0:13: 9                469  0:26: 4                330  0:23:36
289  0:13:33                354  0:11:31                131  0:13:54
422  0:13: 6                230  0:14:21                110  0:12:21
306  0: 7:41                471  0:12:31                188  0:12:46
148  0:13:32                254  0:10:30                428  0:10:47
368  0:10: 9                205  0:14:26                484  0:22:53
114  0: 6:29                294  0:11:41                186  0:13:44
237  0: 7:30                133  0:26:40                224  0:23:41
401  0: 6:22                416  0:13:29                101  0:14:10
232  0:11: 0                288  0:23:23
377  0:13:21                486  0:20:37
```

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
```

```
"""
2017. október: Hiányzások
@author Klemand66
"""


print("\n1. feladat\n")

betxt = open("naplo.txt")

hianyzasok=[]

for sor in betxt:
    darsor=sor.strip().split()
    if darsor[0]=="#":
        ho, nap=int(darsor[1]), int(darsor[2])
    else:
        nev=darsor[0]+" "+darsor[1]
        hianyzas=darsor[2]
        hianyzasok.append([ho, nap, nev, hianyzas])

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A naplóban {len(hianyzasok)} bejegyzés van.')



print("\n3. feladat\n")

igdb, iglandb = 0, 0
for h in hianyzasok:
    igdb+=h[3].count("X")
    iglandb+=h[3].count("I")

print(f'Az igazolt hiányzások száma {igdb}, az igazolatlanoké {iglandb} óra.')



print("\n4. feladat\n")

def hetnapja(honap, nap):
    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)%7
    return napnev[napsorszam]

print(f'A hetnapja függvény:\n\n\
def hetnapja(honap, nap):\n\
    napnev=["vasarnap", "hetfo", "kedd", "szerda", "csutortok", "pentek", "szombat"]\n\
    napszam=[0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 335]\n\
    napsorszam=(napszam[honap-1]+nap)%7\n\
    return napnev[napsorszam]')



print("\n5. feladat\n")

hssz=int(input("Kérem a hónap sorszámát (1<= hónapsorszám <=12): "))
nssz=int(input("Kérem a nap sorszámát (1<= napsorszám <=28,30,31): "))

print(f'Azon a napon {hetnapja(hssz,nssz)} volt.\n')
```

```
print("\n6. feladat\n")

aktnap=input("Kérem a hét egy tanítási napjának a nevét ékezet nélkül: ")
aktora=int(input("Kérem egy óra sorszámát a megadott napon (1 <= sorszám <= 7): "))

akthianyzasok=list(filter(lambda h: hetnapja(h[0],h[1])==aktnap and h[3][aktora-1] in "XI" ,hianyzasok))

print(f'Ekkor összesen {len(akthianyzasok)} óra hiányzás történt.')



print("\n7. feladat\n")

tlista, hlista =[],[] # a tanulók és hiányzásaik listái

for h in hianyzasok:
    hdb=h[3].count("X")+h[3].count("I")
    if h[2] in tlista:
        aktindex=tlista.index(h[2])
        hlista[aktindex]+=hdb
    else:
        tlista.append(h[2])
        hlista.append(hdb)

maxh=max(hlista)
maxindexek=list(filter(lambda i: hlista[i]==maxh ,range(len(tlista))))
maxhianyzok=list(map(lambda i: tlista[i] ,maxindexek))

print(f'A legtöbbet hiányzó tanulók: {", ".join(maxhianyzok)}')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2017okt/EmeltInfo2017okt.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A naplóban 139 bejegyzés van.

3. feladat

Az igazolt hiányzások száma 788, az igazolatlanoké 18 óra.

4. feladat

A hetnapja függvény:

def hetnapja(honap, nap):
    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)%7
    return napnev[napsorszam]

5. feladat

Kérem a hónap sorszámát (1<= hónapsorszám <=12): 2
Kérem a nap sorszámát (1<= napsorszám <=28,30,31): 3
Azon a napon szombat volt.


6. feladat

Kérem a hét egy tanítási napjának a nevét ékezet nélkül: szerda
Kérem egy óra sorszámát a megadott napon (1 <= sorszám <= 7): 3
Ekkor összesen 49 óra hiányzás történt.

7. feladat

A legtöbbet hiányzó tanulók: Kivi Adrienn, Jujuba Ibolya

A befejezéshez nyomd meg az ENTER billentyűt!
```

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.
```

```
"""
2018. május: Társalgó
@author Klemand66
"""


print("\n1. feladat\n")

betxt = open("ajto.txt")

mozgasok, bsz = [], 0

for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[int(darsor[0]),int(darsor[1]),int(darsor[2]),darsor[3]]
            # 0: óra 1: perc 2: azonosító 3: irány (be/ki)
    if darsor[3]=="be":
        bsz+=1
    else:
        bsz-=1
    adatsor.append(bsz)# 4: bentlévők száma
    mozgasok.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

bem=list(filter(lambda m: m[3]=="be" ,mozgasok))
kim=list(filter(lambda m: m[3]=="ki" ,mozgasok))

print(f'Az első belépő: {bem[0][2]}')
print(f'Az utolsó kilépő: {kim[-1][2]}')



print("\n3. feladat\n")

mozgasok.sort(key=lambda m: m[2])

azlista, mdblista = [],[]

for m in mozgasok:
    if m[2] in azlista:
        aktindex=azlista.index(m[2])
        mdblista[aktindex]+=1
    else:
        azlista.append(m[2])
        mdblista.append(1)

kitxt = open("athaladas.txt","w")

for i in range(len(azlista)):
    kitxt.write(f'{azlista[i]} {mdblista[i]}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n4. feladat\n")

bentvoltak=list(filter(lambda i: mdblista[i]%2==1 ,range(len(mdblista))))
bentvoltakstr=list(map(lambda i: str(azlista[i]) ,bentvoltak))

print(f'A végén a társalgóban voltak: {" ".join(bentvoltakstr)}')
```

```
print("\n5. feladat\n")

maxbentlevok=max(mozgasok, key=lambda m: m[4])
print(f'A társalgóban {maxbentlevok[4]} volt a maximális létszám, \
pl. {maxbentlevok[0]}:{maxbentlevok[1]}-kor.')


print("\n6. feladat\n")

aktaz=int(input(f'Kérem egy személy azonosítóját (1 <= azonosító <= {max(azlista)}): '))



print("\n7. feladat\n")

print("A megadott személy a társalgóban tartózkodott:")

aktmozgasok=list(filter(lambda m: m[2]==aktaz ,mozgasok))

for i,a in enumerate(aktmozgasok):
    if i%2==0:
        print(f'{a[0]}:{a[1]}-',end="")
    else:
        print(f'{a[0]}:{a[1]}')
print()



print("\n8. feladat\n")

ido=0
for i,a in enumerate(aktmozgasok):
    aktido=a[0]*60+a[1]
    if i%2==1:
        ido+=aktido
    else:
        ido-=aktido

if len(aktmozgasok)%2==1:
    ido+=15*60
    bef="a társalgóban volt"
else:
    bef="nem volt a társalgóban"

print(f'A(z) {aktaz}. személy összesen {ido} percet volt bent,\n\
a megfigyelés végén {bef}.')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2018maj\EmeltInfo2018maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az első belépő: 2
Az utolsó kilépő: 6

3. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

4. feladat

A végén a társalgóban voltak: 1 11 22 24 29 30 35 37

5. feladat

A társalgóban 12 volt a maximális létszám, pl. 10:44-kor.

6. feladat

Kérem egy személy azonosítóját (1 <= azonosító <= 41): 22

7. feladat

A megadott személy a társalgóban tartózkodott:
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.

A befejezéshez nyomd meg az ENTER billentyűt!
```

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
41 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
```

```
"""
2018. május idegennyelvű: Fogadóóra
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("fogado.txt")

foglalasok=[]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[darsor[0]+" "+darsor[1],darsor[2],darsor[3]]
    # 0: név  1: lefoglalt időpont 2. a foglalás időpontja
    foglalasok.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A foglalások száma: {len(foglalasok)}')



print("\n3. feladat\n")

akttanar=input("Adja meg egy tanár nevét (pl. Nagy Ferenc): ")

aktfoglalasok=list(filter(lambda f: f[0].lower()==akttanar.lower() ,foglalasok))
if len(aktfoglalasok)>0:
    print(f'{aktfoglalasok[0][0]} néven {len(aktfoglalasok)} foglalás van.')
else:
    print("A megadott néven nincs időpontfoglalás.")



print("\n4. feladat\n")

aktido=input("Adjon meg egy érvényes időpontot (pl. 17:10): ")
aktfn="".join(aktido.strip().split(":"))
foglaltak=list(filter(lambda f: f[1]==aktido ,foglalasok))
foglaltak.sort(key=lambda f: f[0])
"""
A fájlban a nevekben, nyilván szándékosan, nincsenek ékezetes karakterek,
ezért a beépített rendezést használtuk, mely az ASCII-kód szerint rendez.
A Bevezető gyakorlatok 2. fejezetének 3. pontjában
szerepel a magyar ábécé szerinti rendezés algoritmusa.
"""

kitxt = open(f'{aktfn}.txt',"w")

for f in foglaltak:
    print(f[0])
    kitxt.write(f'{f[0]}\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n5. feladat\n")

legkorabban=min(foglalasok,key=lambda f: f[2])
print(f'A tanár neve: {legkorabban[0]}')
print(f'A lefoglalt időpont: {legkorabban[1]}')
print(f'A foglalás ideje: {legkorabban[2]}')
```

```
print("\n6. feladat\n")

foglaltak=list(filter(lambda f: f[0]=="Barna Eszter" ,foglalasok))
fidok=[]
for f in foglaltak:
    fidok.append(f[1])

szidok=[]
for o in range(16,18):
    for p in range(0,60,10):
        aktido=f'{o}:{p:02}'
        if aktido not in fidok:
            szidok.append(aktido)

for sz in szidok:
    print(sz)

vszabadok=list(filter(lambda sz: sz>fidok[-1] ,szidok))

print(f'Barna Eszter legkorábban távozhat: {vszabadok[0]}')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2018mid/EmeltInfo2018mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A foglalások száma: 161

3. feladat

Adja meg egy tanár nevét (pl. Nagy Ferenc): nagy ferenc
Nagy Ferenc néven 6 foglalás 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

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

5. feladat

A tanár neve: Csorba Ede
A lefoglalt időpont: 16:30
A 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

A befejezéshez nyomd meg az ENTER billentyűt!


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
```

```
"""
2018. október: Kerítés (Utca)
@author Klemand66
"""

from random import *


print("\n1. feladat\n")

betxt = open("kerites.txt")

keritesek, hszok = [], [0, -1]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[int(darsor[0]),int(darsor[1]),darsor[2]]
    # 0: oldal (páros=0, páratlan=1) 1: szélesség 2: szín (#=nincs befestve :=nem készült el)
    hszok[adatsor[0]]+=2
    adatsor.append(hszok[adatsor[0]]) # 3: házszám
    keritesek.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'Az eladott telkek száma: {len(keritesek)}')



print("\n3. feladat\n")

if keritesek[-1][0]==0:
    p="páros"
else:
    p="páratlan"

print(f'A {p} oldalon adták el az utolsó telket.')
print(f'Az utolsó telek házszáma: {keritesek[-1][3]}')



print("\n4. feladat\n")

paratlanok=list(filter(lambda k: k[3]%2==1 ,keritesek))

jok=list(filter(lambda i: paratlanok[i][2] not in "#:" \
                and paratlanok[i][2]==paratlanok[i+1][2] ,range(len(paratlanok)-1)))


print(f'Az első olyan házszám, ahol a szomszéddal azonos a kerítés színe: {paratlanok[jok[0]][3]}')
```

```
print("\n5. feladat\n")

parosak=list(filter(lambda k: k[3]%2==0 ,keritesek))
maxhsz=[2*len(parosak),2*len(paratlanok)-1]

akthsz=int(input(f'Adjon meg egy házszámot (1 <= házszám <= {maxhsz[0]}/{maxhsz[1]}): '))

hszker=list(filter(lambda k: k[3]==akthsz ,keritesek))
print(f'A kerítés színe/állapota: {hszker[0][2]}')

szinek = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
akthszok=[akthsz]
if akthsz-2>0:
    akthszok.append(akthsz-2)
if akthsz+2<maxhsz[akthsz%2]:
    akthszok.append(akthsz+2)

aktkeritesek=list(filter(lambda k: k[3] in akthszok ,keritesek))
tiltottszinek=list(map(lambda ak: ak[2] ,aktkeritesek))

jo=False
while not jo:
    velszin=choice(szinek)
    if velszin not in tiltottszinek:
        jo=True

print(f'Egy lehetséges festési szín: {velszin}')



print("\n6. feladat\n")

szinsor, hszsor = "", ""
for p in paratlanok:
    szinsor+=p[1]*p[2]
    hszsor+=str(p[3])+(p[1]-len(str(p[3])))*" "

kitxt = open("utcakep.txt","w")

kitxt.write(f'{szinsor}\n')
kitxt.write(f'{hszsor}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2018okt/EmeltInfo2018okt.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az eladott telkek száma: 98

3. feladat

A páros oldalon adták el az utolsó telket.
Az utolsó telek házszáma: 78

4. feladat

Az első olyan házszám, ahol a szomszéddal azonos a kerítés színe: 73

5. feladat

Adjon meg egy házszámot (1 <= házszám <= 78/117): 83
A kerítés színe/állapota: A
Egy lehetséges festési szín: V

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!


Az utcakep.txt szövegfile:

KKKKKKKK:::::::::::SSSSSSSSSBBBBBBBBFFFFFFFFFKKKKKKKKKKIIIIIIIIIDDDDDDDDDDDDDDDDDSSSSSSSSSSSSSSSSPPPPPPPPPPPPPPPPKKK
1       3         5        7       9        11        13      15              17             19             21
KKKKKKKKKKKKKKKK:::::::::::::::::::::::::::::IIIIIIIIIIIIIIIIIIYYYYYYYYYYYYYYYYYYYYCCCCCCCCCCIIIIIIIIIWWWWWWWWWWWWW
                23       25               27               29                  31       33      35
WLLLLLLLLLLLLLLLLLYYYYYYYYYYYYYYYYYYYNNNNNNNNNNNNTTTTTTTTTTTTTTTTTTQQQQQQQQQQQQQQQQQQ:::::::::::::::::HHHHHHHHHHH
 37               39                 41         43               45               47              49
HHHHHHHFFFFFFFFFFFFFFFWWWWWWWWWWWWWWXXXXXXXXXXXXXXXXXXXXXX##############################HHHHHHHHHHHHHHHHHHHDDDD
       51             53            55                  57           59                  61                 63
DDDDDDDDDDDDDDPPPPPPPPPUUUUUUUUUUUUUXXXXXXXXXXXXXXDDDDDDDDBBBBBBBBBBBBBBBBBBBBBBOOOOOOOOOOOOOOOOOOOO::::::::::::::
              65       67          69           71      73             75      77                 79
:CCCCCCCCCCCCCCCCAAAAAAAABBBBBBBBBBBBBBUUUUUUUUUUUUUUUURRRRRRRRRSSSSSSSSSSSSSSSSSIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
 81              83      85            87             89       91               93             95
IIIVVVVVVVVVVVVUUUUUUUUUUUUUUUUBBBBBBBBBBBBBBBBCCCCCCCCCCCMMMMMMMMMMMMMMM:::::::::::QQQQQQQQQQQQCCCCCCCCCCCCCCCCCCC
   97         99             101            103       105            107        109         111
CVVVVVVVVVVVVZZZZZZZZZZZZZZZZZZZZZZVVVVVVVVVVVVVVVVVVVV
 113         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
```

```
"""
2019. május: Céges autók
@author Klemand66
"""


print("\n1. feladat\n")

betxt = open("autok.txt")

autok=[]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[int(darsor[0]),darsor[1],darsor[2],int(darsor[3]),int(darsor[4]),int(darsor[5])]
    #        0: nap        1: óó:pp   2: rsz    3: azonosító   4: km          5: ki/be=0/1
    autok.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

kivittek=list(filter(lambda a: a[5]==0 ,autok))
print(f'{kivittek[-1][0]}. nap rendszám: {kivittek[-1][2]}')



print("\n3. feladat\n")

aktnap=int(input(f'Kérem egy nap sorszámát (1 <= nap <== {autok[-1][0]}): '))

aktautok=list(filter(lambda a: a[0]==aktnap ,autok))
print(f'Forgalom a(z) {aktnap}. napon:')

for a in aktautok:
    if a[5]==0:
        ir="ki"
    else:
        ir="be"
    print(f'{a[1]} {a[2]} {a[3]} {ir} ')



print("\n4. feladat\n")

kintvannak=[] # tudjuk, hogy 10 autó van és kezdetben mind bent van
for i in range(10):
    kintvannak.append(False)

for a in autok:
    kintvannak[int(a[2][-1])]=a[5]==0

print(f'A hónap végén {kintvannak.count(True)} autót nem hoztak vissza.')



print("\n5. feladat\n")

for i in range(10):
    aktautok=list(filter(lambda a: int(a[2][-1])==i ,autok))
    print(f'CEG30{i} {aktautok[-1][4]-aktautok[0][4]} km')
```

```
print("\n6. feladat\n")

maxutak=[]
for i in range(10):
    aktautok=list(filter(lambda a: int(a[2][-1])==i ,autok))
    aktmaxind=max(range(1,len(aktautok)), key=lambda i: aktautok[i][4]-aktautok[i-1][4])
    maxutak.append([aktautok[aktmaxind][4]-aktautok[aktmaxind-1][4],aktautok[aktmaxind][3]])

maxut=max(maxutak, key=lambda m: m[0])

print(f'A leghosszabb út: {maxut[0]} km, a személy: {maxut[1]}')



print("\n7. feladat\n")


aktrsz=input("Kérek egy rendszámot: ")

fnev=f'{aktrsz}menetlevel.txt'
kitxt = open(fnev,"w")

aktautok=list(filter(lambda a: a[2]==aktrsz ,autok))

for a in aktautok:
    if a[5]==0:
        kitxt.write(f'{a[3]}\t{a[0]:2}.\t{a[1]}\t{a[4]:5} km\t')
    else:
        kitxt.write(f'{a[0]:2}.\t{a[1]}\t{a[4]:5} km\t\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2019maj/EmeltInfo2019maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

30. nap rendszám: CEG300

3. feladat

Kérem egy nap sorszámát (1 <= nap <== 30): 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

A leghosszabb út: 1551 km, a személy: 506

7. feladat

Kérek egy rendszámot: CEG304

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!


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 km11.     21:20   11999 km
574     13.     10:12   11999 km13.     20:17   12153 km
549     15.     14:46   12153 km16.     21:29   12854 km
591     18.     08:43   12854 km18.     18:54   13007 km
534     19.     07:53   13007 km19.     22:43   13279 km
560     20.     08:35   13279 km20.     11:27   13452 km
588     21.     16:58   13452 km23.     20:28   14335 km
512     24.     16:58   14335 km26.     22:21   15041 km
504     27.     13:47   15041 km
```

2019. május, idegennyelvű: 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.
```

```
"""
2019. május idegennyelvű: Tantárgyfelosztás
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("beosztas.txt")

beosztasok=[]

for sor in betxt:
    nev=sor.strip()                     # 0
    tt=next(betxt).strip()              # 1
    oszt=next(betxt).strip()            # 2
    oraszam=int(next(betxt).strip())    # 3
    beosztasok.append([nev,tt,oszt,oraszam])

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A fájlban {len(beosztasok)} bejegyzés van.')



print("\n3. feladat\n")

oszamok=list(map(lambda i: beosztasok[i][3] ,range(len(beosztasok))))

print(f'Az iskolában a heti összóraszám {sum(oszamok)}')



print("\n4. feladat\n")

akttanar=input("Kérem egy tanár nevét (Pl. Albatrosz Aladin): ").strip()
aktbeosztasok=list(filter(lambda b: b[0]==akttanar  ,beosztasok))
aktoszamok=list(map(lambda i: aktbeosztasok[i][3] ,range(len(aktbeosztasok))))

print(f'A megadott tanár heti óraszáma: {sum(aktoszamok)}')



print("\n5. feladat\n")

ofok=list(filter(lambda b: b[1]=="osztalyfonoki"  ,beosztasok))

kitxt = open("of.txt","w")

for o in ofok:
    kitxt.write(f'{o[2]:>4} - {o[0]}\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.")
```

```
print("\n6. feladat\n")

aktoszt=input("Kérem egy osztály nevét (Pl. 10.b): ").strip()
akttt=input("Kérem egy tantárgy nevét (Pl. kemia): ").strip()
aktbeosztasok=list(filter(lambda b: b[2]==aktoszt and b[1]==akttt ,beosztasok))

if len(aktbeosztasok)>1:
    print("Csoportbontásban tanulják.")
else:
    print("Osztályszinten tanulják.")



print("\n7. feladat\n")

tanarlista=[]
for b in beosztasok:
    if b[0] not in tanarlista:
        tanarlista.append(b[0])

print(f'Az iskolában {len(tanarlista)} tanár tanít.')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2019mid/EmeltInfo2019mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A fájlban 329 bejegyzés van.

3. feladat

Az iskolában a heti összóraszám 1016

4. feladat

Kérem egy tanár nevét (Pl. Albatrosz Aladin): Albatrosz Aladin
A megadott tanár heti óraszáma: 24

5. feladat


A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

6. feladat

Kérem egy osztály nevét (Pl. 10.b): 10.b
Kérem egy tantárgy nevét (Pl. kemia): kemia
Csoportbontásban tanulják.

7. feladat

Az iskolában 49 tanár tanít.

A befejezéshez nyomd meg az ENTER billentyűt!



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 járművőn 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
...
```

```
"""
2019. október: eUtazás
@author Klemand66
"""


print("\n1. feladat\n")

betxt = open("utasadat.txt")

utasok=[]
for sor in betxt:
    darsor=sor.strip().split()
    megallo=int(darsor[0])
    dat=darsor[1][:8]
    az=darsor[2]
    tip=darsor[3]
    erv=darsor[4]
    utasok.append([megallo,dat,az,tip,erv])

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A buszra {len(utasok)} utas akart felszállni.')



print("\n3. feladat\n")

ervenytelenek=list(filter(lambda u: (u[3]=="JGY" and u[4]<"1") or (u[3]!="JGY" and u[4]<u[1]) ,utasok))

print(f'A buszra {len(ervenytelenek)} utas nem szállhatott fel.')



print("\n4. feladat\n")

felszallok=[]
for i in range(30):
    felszallok.append(len(list(filter(lambda u: u[0]==i ,utasok))))

legtobben=max(felszallok)
maxmegallo=felszallok.index(legtobben)

print(f'A legtöbb utas ({legtobben} fő) a {maxmegallo}. megállóban próbált felszállni.')



print("\n5. feladat\n")

ervenyesek=list(filter(lambda u: (u[3]=="JGY" and u[4]>"0") or (u[3]!="JGY" and u[4]>=u[1]) ,utasok))
ingyenesek=list(filter(lambda u: u[3]=="NYP" or u[3]=="RVS" or u[3]=="GYK",ervenyesek))
kedvesek=list(filter(lambda u: u[3]=="TAB" or u[3]=="NYB" ,ervenyesek))

print(f'Ingyenesen utazók száma: {len(ingyenesek)} fő')
print(f'Kedvezményesen utazók száma: {len(kedvesek)} fő')
```

```
print("\n6. feladat\n")

def napokszama(e1,h1,n1,e2,h2,n2):
    h1=(h1+9)%12
    e1=e1-h1//10
    d1=365*e1 + e1//4 - e1//100 + e1//400 + (h1*306+5)//10 + n1 - 1
    h2=(h2+9)%12
    e2=e2-h2//10
    d2=365*e2 + e2//4 - e2//100 + e2//400 + (h2*306+5)//10 + n2 - 1
    return d2-d1

print(f'Az elkészített függvény:\n\n\
def napokszama(e1,h1,n1,e2,h2,n2):\n\
    h1=(h1+9)%12\n\
    e1=e1-h1//10\n\
    d1=365*e1 + e1//4 - e1//100 + e1//400 + (h1*306+5)//10 + n1 - 1\n\
    h2=(h2+9)%12\n\
    e2=e2-h2//10\n\
    d2=365*e2 + e2//4 - e2//100 + e2//400 + (h2*306+5)//10 + n2 - 1\n\
    return d2-d1')



print("\n7. feladat\n")

kitxt = open("figyelmeztetes.txt","w")

berletesek=list(filter(lambda u: u[3]!="JGY" ,utasok))

for b in berletesek:
    e1, h1, n1 = int(b[1][:4]), int(b[1][4:6]), int(b[1][6:8])
    e2, h2, n2 = int(b[4][:4]), int(b[4][4:6]), int(b[4][6:8])
    if napokszama(e1,h1,n1,e2,h2,n2)<=3:
        kitxt.write(f'{b[2]} {e2:04}-{h2:02}-{n2:02}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:/Python/KlEmProgPy/EmeltInfo2019okt/EmeltInfo2019okt.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

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
Kedvezményesen utazók száma: 200 fő

6. feladat

Az elkészített függvény:

def napokszama(e1,h1,n1,e2,h2,n2):
    h1=(h1+9)%12
    e1=e1-h1//10
    d1=365*e1 + e1//4 - e1//100 + e1//400 + (h1*306+5)//10 + n1 - 1
    h2=(h2+9)%12
    e2=e2-h2//10
    d2=365*e2 + e2//4 - e2//100 + e2//400 + (h2*306+5)//10 + n2 - 1
    return d2-d1

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!


A figyelmeztetes.txt szövegfile:

2020534 2019-03-28                    2223555 2019-03-27                    3600757 2019-03-27
2416208 2019-03-29                    3134404 2019-03-01                    2190408 2019-03-27
3199016 2019-03-28                    2258830 2019-03-26                    3680746 2019-03-27
1446402 2019-03-27                    3023275 2019-03-29                    2118150 2019-03-25
4272667 2019-03-29                    4033629 2019-03-25                    2083298 2019-03-25
2734415 2019-03-27                    2960983 2019-03-26                    4658969 2019-03-27
2416648 2019-03-27                    1581897 2019-03-27                    4669550 2019-03-28
4043370 2019-03-26                    2761792 2019-03-28                    4108241 2019-03-26
4159578 2019-03-28                    2946924 2019-03-25                    4299308 2019-03-27
2896921 2019-03-27                    3488169 2019-03-25                    2373593 2019-03-29
3972945 2019-03-26                    2416253 2019-03-25                    1720477 2019-03-28
4764322 2019-03-27                    3702373 2019-03-26                    3380452 2019-03-27
1473856 2019-03-29                    3173542 2019-03-28                    3677994 2019-03-27
1618545 2019-03-26                    2496259 2019-03-29                    3552479 2019-03-27
4400156 2019-03-27                    3868285 2019-03-25
2132147 2019-03-25                    4688169 2019-03-26
1290376 2019-03-25                    1062584 2019-03-28
```

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 adatait tartalmazza. 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 ########
```

```
"""
2020. május: Meteorológiai jelentés
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("tavirathu13.txt")

meresek=[]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[darsor[0],darsor[1][:2],darsor[1][2:],darsor[2][:3],int(darsor[2][3:]),int(darsor[3])]
    #        0: kód  1: óra        2: perc       3: szélirány  4: szélerősség     5: hőmérséklet
    meresek.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

aktkod=input("Adja meg egy település kódját (Pl. BP, DC, PP, SM)! Település: ").upper()
amek=list(filter(lambda m: m[0]==aktkod ,meresek)) # az aktkódhoz tartozó mérések

print(f'Az utolsó mérési adat a megadott településről {amek[-1][1]}:{amek[-1][2]}-kor érkezett.')



print("\n3. feladat\n")

minh=min(meresek,key=lambda m: m[5])
maxh=max(meresek,key=lambda m: m[5])

print(f'A legalacsonyabb hőmérséklet: {minh[0]} {minh[1]}:{minh[2]} {minh[5]} fok.')
print(f'A legmagasabb hőmérséklet: {maxh[0]} {maxh[1]}:{maxh[2]} {maxh[5]} fok.')



print("\n4. feladat\n")

szelcsendesek=list(filter(lambda m: m[3]=="000" and m[4]==0 ,meresek))

if len(szelcsendesek)>0:
    for sz in szelcsendesek:
        print(f'{sz[0]} {sz[1]}:{sz[2]}')
else:
    print("Nem volt szélcsend a mérések idején.")



print("\n5. feladat\n")

vlista=[]
for m in meresek:
    if m[0] not in vlista:
        vlista.append(m[0])

for v in vlista:
    aktvarosok=list(filter(lambda m: m[0]==v ,meresek))
    print(v,end=" ")

    aktidok=list(filter(lambda a: a[1] in ["01","07","13","19"] ,aktvarosok))
    # Előfordulhat, hogy az egyik órában több adat is jött, a másikban egy sem.
    oralista=[]
    for a in aktidok:
        if a[1] not in oralista:
            oralista.append(a[1])
    if len(oralista)==4:
        homersekletek=list(map(lambda i: aktidok[i][5] ,range(len(aktidok))))
        print(f'Középhőmérséklet: {sum(homersekletek)/len(aktidok):.0f};',end=" ")
    else:
        print("NA;",end=" ")

    minh=min(aktvarosok,key=lambda a: a[5])
    maxh=max(aktvarosok,key=lambda a: a[5])
    print(f'Hőmérséklet-ingadozás: {maxh[5]-minh[5]}')
```

```
print("\n6. feladat\n")

for v in vlista:
    kitxt = open(f'{v}.txt',"w")
    kitxt.write(f'{v}\n')
    aktvarosok=list(filter(lambda m: m[0]==v ,meresek))
    for a in aktvarosok:
        kitxt.write(f'{a[1]}:{a[2]} {a[4]*"#"}\n')
    kitxt.close()

print("A kiíratás és a szövegfájlok lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2020maj\EmeltInfo2020maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Adja meg egy település kódját (Pl. BP, DC, PP, SM)! Település: bp
Az utolsó mérési adat a megadott településről 23:30-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 kiíratás és a szövegfájlok lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A BC.txt szövegfile:

```
BC
00:45 ###
01:45 ####
02:45 ######
03:45 ##
04:45
05:45 ####
11:45
17:45 ########
```

A BP.txt szövegfile:

```
BP
00:00 ##
00:30 ##
01:00
01:30 ##############
02:00 ##################
02:30 ###########
03:00 #######
03:30 #
04:00 #
04:30 #####
05:00 #######
05:30 ########
06:00 #########
06:30 #######
07:00 ###########
07:30 ########
08:00 ########
08:30 #########
09:00 #######
09:30 ##########
10:00 ###############
10:30 ################
11:00 ###############
11:30 ###############
12:00 #############
12:30 ################
13:00 ###############
13:30 ##################
14:00 #############
14:30 ###########
15:00 ###############
15:30 ################
16:00 ###########
16:30 ###########
17:00 #########
17:30 ########
17:48 #######
18:00 #######
18:30 ######
19:00 ######
19:30 ########
19:39 #######
19:44 #######
20:00 ##########
20:30 ########
21:00 #####
21:14 #
21:30 #
22:00 ###
22:09 ###
22:30 #####
22:42 #####
23:00 ########
23:30 ####

...
```

Az SN.txt szövegfile:

```
SN
00:15 ####
00:45 ######
01:15 ######
01:45 ####
02:15 ####
02:45 ####
03:15
03:45 ####
04:15 ######
04:45 ##
05:15
05:45
06:15 ####
06:45 ######
07:15 ######
07:45 ######
08:15 ######
08:45 ####
09:15 ####
09:45 ##
10:15 ####
10:45 ####
11:15 ####
11:45 ######
12:15 ######
12:45 ######
13:15 ######
13:45 ###############
14:15 ############
14:45 ###############
15:15 ###############
15:45 ############
16:15 ##########
16:45 ##########
17:15 ########
17:45 ####
18:15 ##########
18:45 ##########
19:15 ######
19:45 ######
20:15 ####
20:45 ######
21:15 ########
21:45 ######
22:15 ########
22:29 ########
22:45 ########
23:23 ######
23:53 ########
```

2020. május, idegennyelvű: 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
```

```
"""
2020. május idegennyelvű: Menetrend
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("vonat.txt")

vonatok=[]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[int(darsor[0]),int(darsor[1]),int(darsor[2])*60+int(darsor[3]),darsor[4]]
           # 0: vonat       1: állomás     2: idő percben                  3: E/I
    vonatok.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

allomasdb, vonatdb = vonatok[-1][1]+1, vonatok[-1][0]
print(f'Az állomások száma: {allomasdb}') # a kiinduló állomás a 0
print(f'A vonatok száma: {vonatdb}')      # az első vonat az 1



print("\n3. feladat\n")

maxallasok=[]
for i in range(1,vonatdb+1):
    ind=list(filter(lambda v: v[0]==i and v[3]=="I" ,vonatok))
    erk=list(filter(lambda v: v[0]==i and v[3]=="E" ,vonatok))
    maxi=max(range(1,len(ind)), key=lambda j: ind[j][2]-erk[j-1][2])
    maxallasok.append([i, maxi, ind[maxi][2]-erk[maxi-1][2]])

maxallas=max(maxallasok, key=lambda m: m[2])

print(f'A legtöbbet álló vonat: \
a(z) {maxallas[0]}. vonat a(z) {maxallas[1]}. állomáson {maxallas[2]} percet állt.')



print("\n4. feladat\n")

vaz=int(input(f'Kérem egy vonat azonosítóját (0 <= azonosító <= 20): '))
darido=input(f'Adjon meg egy időpontot óra perc alakban: ').strip().split()
aktido = int(darido[0])*60 + int(darido[1])



print("\n5. feladat\n")

aktvonatok=list(filter(lambda v: v[0]==vaz ,vonatok))
menetido=2*60+22
aktmenetido=aktvonatok[-1][2] - aktvonatok[0][2]
elteres=aktmenetido-menetido

if elteres<0:
    print(f'A(z) {vaz}. vonat útja {abs(elteres)} perccel rövidebb volt az előírtnál.')
elif elteres==0:
    print(f'A(z) {vaz}. vonat útja pontosan az előírt ideig tartott.')
else:
    print(f'A(z) {vaz}. vonat útja {elteres} perccel hosszabb volt az előírtnál.')
```

```
print("\n6. feladat\n")

kitxt = open(f'halad{vaz}.txt',"w")

akterkezesek=list(filter(lambda v: v[0]==vaz and v[3]=="E" ,vonatok))

for v in akterkezesek:
    kitxt.write(f'{v[1]:2}. állomás: {v[2]//60:02}:{v[2]%60:02}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n7. feladat\n")

for i in range(1,vonatdb+1):
    ind=list(filter(lambda v: v[0]==i and v[3]=="I" and aktido>=v[2] ,vonatok))
    erk=list(filter(lambda v: v[0]==i and v[3]=="E" and aktido>=v[2] ,vonatok))
    if len(ind)>0 and len(erk)<10: # ha úton volt
        if len(ind)==len(erk):
            print(f'A(z) {i}. vonat a {len(erk)}. állomáson állt.')
        else:
            print(f'A(z) {i}. vonat a {len(ind)-1}. és a {len(ind)}. állomás között járt.')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2020mid\EmeltInfo2020mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az állomások száma: 11
A vonatok száma: 12

3. feladat

A legtöbbet álló vonat: a(z) 5. vonat a(z) 6. állomáson 10 percet állt.

4. feladat

Kérem egy vonat azonosítóját (0 <= azonosító <= 20): 2
Adjon meg egy időpontot óra perc alakban: 7 16

5. feladat

A(z) 2. vonat útja 2 perccel hosszabb volt az előírtnál.

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

7. feladat

A(z) 1. vonat a 6. állomáson állt.
A(z) 2. vonat a 2. és a 3. állomás között járt.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A halad2.txt szövegfile:

```
 1. állomás: 06:58
 2. állomás: 07:11
 3. állomás: 07:31
 4. állomás: 07:48
 5. állomás: 07:59
 6. állomás: 08:11
 7. állomás: 08:45
 8. állomás: 08:51
 9. állomás: 09:00
10. állomás: 09:09
```

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.

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.

Például:

```
…
2018.01.19
Puzzles
3x10
43
0
NI
Puzzles
3x11
43
0
…
```

- A dátumokat mindig „éééé.hh.nn” formátumban rögzítették. Vannak olyan sorozatrészek, amelyeknek a lista rögzítésekor még nem tudták az adásba kerülésük idejét. Ezeknél a dátum helyett 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
```

```
"""
2020. október: Sorozatok
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("lista.txt")

sorozatok=[]
for sor in betxt:
    datum=sor.strip()
    cim=next(betxt).strip()
    evxep=next(betxt).strip()
    ephossz=int(next(betxt).strip())
    latta=int(next(betxt).strip()) # 0: nem látta 1: látta
    sorozatok.append([datum, cim, evxep, ephossz, latta])

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

ismertek=list(filter(lambda s: s[0]!="NI",sorozatok))
print(f'A listában {len(ismertek)} db vetítési dátummal rendelkező epizód van.')



print("\n3. feladat\n")

latta=list(filter(lambda s: s[4]==1 ,sorozatok))
print(f'A listában lévő epizódok {100*len(latta)/len(sorozatok):.2f}%-át látta.')



print("\n4. feladat\n")

hosszak=list(map(lambda i: latta[i][3], range(len(latta))))
osszhossz=sum(hosszak)
perc, ora = osszhossz%60, osszhossz//60
nap, ora = ora//24, ora%24

print(f'Sorozatnézéssel {nap} napot {ora} órát és {perc} percet töltött.')



print("\n5. feladat\n")

aktdat=input("Adjon meg egy dátumot (pl. 2017.10.18)! Dátum = ")
addignem=list(filter(lambda s: s[0]<=aktdat and s[4]==0 ,sorozatok))
for an in addignem:
   print(f'{an[2]} {an[1]}')



print("\n6. feladat\n")

def hetnapja(ev, ho, nap):
    napok=["v", "h", "k", "sze", "cs", "p", "szo"]
    honapok=[0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4]
    if ho<3:
        ev-=1
    return napok[(ev + ev//4 - ev//100 + ev//400 + honapok[ho-1] + nap)%7]

print(f'Az elkészített függvény:\n\n\
def hetnapja(ev, ho, nap):\n\
    napok=["v", "h", "k", "sze", "cs", "p", "szo"]\n\
    honapok=[0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4]\n\
    if ho<3:\n\
        ev-=1\n\
    return napok[(ev + ev//4 - ev//100 + ev//400 + honapok[ho-1] + nap)%7]')
```

```
print("\n7. feladat\n")

aktnap=input("Adja meg a hét egy napját (h k sze cs p szo v)! Nap = ").strip()
print("\nA következő sorozatokat vetítik a hét megadott napján:")
aktnapon=list(filter(lambda s: \
         s[0]!="NI" and hetnapja(int(s[0][:4]),int(s[0][5:7]),int(s[0][8:]))==aktnap ,sorozatok))

aktfilmlista=[]
for a in aktnapon:
    if a[1] not in aktfilmlista:
        aktfilmlista.append(a[1])

if len(aktfilmlista)>0:
    for f in aktfilmlista:
        print(f)
else:
    print("Az adott napon nem kerül adásba sorozat.")



print("\n8. feladat\n")

kitxt = open("summa.txt","w")

filmlista=[]
for s in sorozatok:
    if s[1] not in filmlista:
        filmlista.append(s[1])

for f in filmlista:
    aktfilmek=list(filter(lambda s: s[1]==f ,sorozatok))
    kitxt.write(f'{f} {len(aktfilmek)*aktfilmek[0][3]} {len(aktfilmek)}\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2020okt\EmeltInfo2020okt.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

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 (pl. 2017.10.18)! Dátum = 2017.10.18
7x01 The Fable
7x02 The Fable
15x04 Military Police
5x03 Spy School
5x04 Spy School
4x04 The Elite Minds

6. feladat

Az elkészített függvény:

def hetnapja(ev, ho, nap):
    napok=["v", "h", "k", "sze", "cs", "p", "szo"]
    honapok=[0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4]
    if ho<3:
        ev-=1
    return napok[(ev + ev//4 - ev//100 + ev//400 + honapok[ho-1] + nap)%7]

7. feladat

Adja meg a hét egy napját (h k sze cs p szo v)! Nap = cs

A következő sorozatokat vetítik a hét megadott napján:
The Hospital
Spectacular Power
Upper Story
Chicago Flame
Shrinktime

8. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A summa.txt szövegfile:

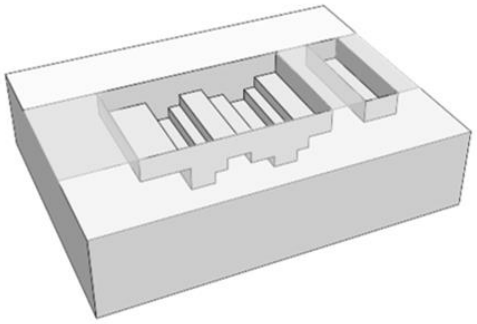
```
Games 420 7
The Fable 588 14
The IT Guy 450 10
The Hospital 378 9
The Killers 504 12
Maniac Man 630 15
Adventures with Eric 450 10
Military Police 480 12
The Streets of LA 520 13
Puzzles 688 16
Spy School 495 11
Spectacular Power 588 14
Upper Story 286 13
Chicago Flame 462 11
The Elite Minds 672 16
Shrinktime 945 35
Perfectly Numb 60 1
```

2021. május: Gödrök

Informatika emelt szint | Azonosító jel:

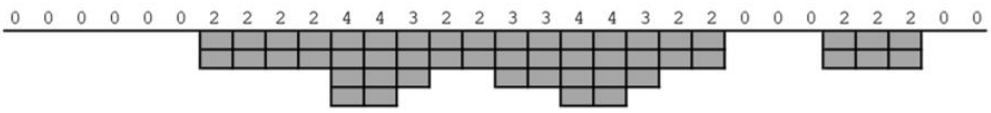
4. Gödrök

Egy teljesen sík terepen a talaj olyan anyagból van, ami nem ereszti át a vizet. Ezen a területen egy egyenes mentén munkagépekkel 10 méter széles csatornát építenek. A munka még nem készült el, ezért a csatorna mélysége nem állandó, helyenként a felszín is érintetlen. A már elkészült résszel, mint különálló gödrök sorozatával foglalkozik a feladat. Az egyszerűbb kezelés érdekében a gödröket úgy tekintjük, hogy oldalfaluk függőleges, teljes szélességben azonos mélységű, így a keresztmetszeti kép jól leírja a terepviszonyokat.



A `melyseg.txt` fájlban méterenként rögzítették, hogy azon a szakaszon milyen mélyen van a gödör alja. Minden sor egy-egy egész számot tartalmaz, amely a mélység értékét mutatja – szintén méterben. A fájl legfeljebb 2000 számot tartalmaz, értékük legfeljebb 30 lehet. Tudjuk, hogy az első és az utolsó méteren sértetlen a felszín, tehát ott biztosan a 0 szám áll.

Például:



A fenti példában látható keresztmetszeti képen a `melyseg.txt` bemeneti fájl tartalmának a kezdete látható. Az egyszerűbb szemléltetés miatt a forrásfájlban külön sorokban szereplő értékeket itt az ábrán egymás mellett szerepeltetjük. Leolvasható, hogy az első méteren 0 a mélység. Az első gödör a 7. méteren kezdődik. Az első gödör 16 méter hosszan tart, legnagyobb mélysége 4 méter, térfogata 440 m^3. A második gödör a 26. méternél kezdődik, 3 méter hosszan tart, térfogata 60 m^3.

Készítsen programot, amely a `melyseg.txt` állományt felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `godor` 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 `melyseg.txt` fájl tartalmát! Írja ki a képernyőre, hogy az adatforrás hány adatot tartalmaz!
2. Olvasson be egy távolságértéket, majd írja a képernyőre, hogy milyen mélyen van a gödör alja azon a helyen! Ezt a távolságértéket használja majd a 6. feladat megoldása során is!
3. Határozza meg, hogy a felszín hány százaléka maradt érintetlen és jelenítse meg 2 tizedes pontossággal!
4. Írja ki a `godrok.txt` fájlba a gödrök leírását, azaz azokat a számsorokat, amelyek egy-egy gödör méterenkénti mélységét adják meg! Minden gödör leírása külön sorba kerüljön! Az állomány pontosan a gödrök számával egyező számú sort tartalmazzon!

Informatika emelt szint | Azonosító jel:

A *godrok.txt* fájl első két sorának tartalma:

```
2 2 2 2 4 4 3 2 2 3 3 4 4 3 2 2
2 2 2
…
```

5. Határozza meg a gödrök számát és írja a képernyőre!
6. Ha a 2. feladatban beolvasott helyen nincs gödör, akkor „Az adott helyen nincs gödör.” üzenetet jelenítse meg, ha ott gödör található, akkor határozza meg, hogy
   a) mi a gödör kezdő és végpontja! A meghatározott értékeket írja a képernyőre! (Ha nem tudja meghatározni, használja a további részfeladatoknál a 7 és 22 értéket, mint a kezdő és a végpont helyét)
   b) a legmélyebb pontja felé mindkét irányból folyamatosan mélyül-e! Azaz a gödör az egyik szélétől monoton mélyül egy pontig, és onnantól monoton emelkedik a másik széléig. Az eredménytől függően írja ki a képernyőre a „Nem mélyül folyamatosan.” vagy a „Folyamatosan mélyül.” mondatot!
   c) mekkora a legnagyobb mélysége! A meghatározott értéket írja a képernyőre!
   d) mekkora a térfogata, ha szélessége minden helyen 10 méternyi! A meghatározott értéket írja a képernyőre!
   e) a félkész csatorna esőben jelentős mennyiségű vizet fogad be. Egy gödör annyi vizet képes befogadni anélkül, hogy egy nagyobb szélvihar hatására se öntsön ki, amennyi esetén a víz felszíne legalább 1 méter mélyen van a külső felszínhez képest. Írja a képernyőre ezt a vízmennyiséget!

Minta a szöveges kimenetek kialakításához:

```
1. feladat
A fájl adatainak száma: 694

2. feladat
Adjon meg egy távolságértéket! 9
Ezen a helyen a felszín 2 méter mélyen van.

3. feladat
Az érintetlen terület aránya 10.09%.

5. feladat
A gödrök száma: 22

6. feladat
a)
A gödör kezdete: 7 méter, a gödör vége: 22 méter.
b)
Nem mélyül folyamatosan.
c)
A legnagyobb mélysége 4 méter.
d)
A térfogata 440 m^3.
e)
A vízmennyiség 280 m^3.
```

45 pont

```
"""
2021. május: Gödrök
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("melyseg.txt")

melysegek=[]
for sor in betxt:
    melysegek.append(sor.strip())

betxt.close()
print(f'A fájl adatainak száma: {len(melysegek)}')



print("\n2. feladat\n")

akttav=int(input(f'Adjon meg egy távolságértéket (1 és {len(melysegek)} közötti egész számot): '))
print(f'Ezen a helyen a felszín {melysegek[akttav-1]} méter mélyen van.')



print("\n3. feladat\n")

print(f'Az érintetlen terület aránya {100*melysegek.count("0")/len(melysegek):.2f}%.')



print("\n4. feladat\n")

kitxt = open("godrok.txt","w")

darmelysegek=(" "+" ".join(melysegek)).split(" 0")
# Nem darabolhatok simán a "0"-val, a mélység is lehet "10".
godrok=list(filter(lambda d: d.strip()!="" ,darmelysegek))

for g in godrok:
    kitxt.write(f'{g.strip()}\n') # a gödrök elején szóköz maradt

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n5. feladat\n")

print(f'A gödrök száma: {len(godrok)}')
```

```
print("\n6. feladat\n")

if melysegek[akttav]>"0":
    fgelott=list(filter(lambda i: i<akttav and melysegek[i]=="0" ,range(len(melysegek))))
    fgutan=list(filter(lambda i: i>akttav and melysegek[i]=="0" ,range(len(melysegek))))
    gkezd, gvege =(fgelott[-1]+1)+1, (fgutan[0]+1)-1
    print(f'a)\nA gödör kezdete: {gkezd} méter, a gödör vége: {gvege} méter.')
    godor=list(map(lambda m: int(m) ,melysegek[gkezd-1:gvege]))
    maxm = max(godor)
    maxi = godor.index(maxm)
    folyamatos=list(map(lambda i: (i<=maxi and godor[i]>=godor[i-1]) \
              or (i>maxi and godor[i]<=godor[i-1]),range(1,len(godor))))
    if (folyamatos).count(False)==0:
        print(f'b)\nFolyamatosan mélyül.')
    else:
        print(f'b)\nNem mélyül folyamatosan.')
    print(f'c)\nA legnagyobb mélysége: {maxm} méter.')
    terfogat=sum(list(map(lambda melyseg: melyseg*10 ,godor)))
    print(f'd)\nA térfogata: {terfogat} m^3.')
    print(f'e)\nA vízmennyiség: {terfogat-10*(gvege-gkezd+1)} m^3.')
else:
    print("Az adott helyen nincs gödör.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2021maj\EmeltInfo2021maj.py ==================

1. feladat

A fájl adatainak száma: 694

2. feladat

Adjon meg egy távolságértéket (1 és 694 közötti egész számot): 9
Ezen a helyen a felszín 2 méter mélyen van.

3. feladat

Az érintetlen terület aránya 10.09%.

4. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

5. feladat

A gödrök száma: 22

6. feladat

a)
A gödör kezdete: 7 méter, a gödör vége: 22 méter.
b)
Nem mélyül folyamatosan.
c)
A legnagyobb mélysége: 4 méter.
d)
A térfogata: 440 m^3.
e)
A vízmennyiség: 280 m^3.

A befejezéshez nyomd meg az ENTER billentyűt!


A godrok.txt szövegfile:

2 2 2 2 4 4 3 2 2 3 3 4 4 3 2 2
2 2 2
2 1 3 1 1 1 1 3 3 3 4 2 4 4 4 3 3 3 1
1
1 1
1 1 2 1
2 3 3 3 3 5 5 6 5 6 7 8 8 7 9 9 9 8 8 8 9 8 8 8 6 5 5 5 4 2 4 3 4 5 7 7 8 9 9 8 6 4 6 8 8 9 11 9 9 11 10 9 ...
1 3 2 3 3 3 3 4 5 6 6 4 2 2 2
2 2 4 6 4 6 5 7 6 5 3 4 2 4 6 8 7 8 7 9 10 11 9 11 12 11 9 9 10 8 10 10 10 9 8 7 7 7 7 6 4 3 3 4 2 2 4 4 2 ...
1 1 2
2 3 3 1 1 2 2 2 1 1
1 1 1
1
1
2 2 2
1
2 2 4 4 2 1
1 2 1 2 1 3 1
1 1 3 4 5 3 3 4 5 7 8 10 10 8 10 10 11 11 11 9 8 10 9 7 9 11 12 12 12 12 10 9 9 11 12 12 14 12 13 15 17 17 ...
1 1 1 1 3 5 7 7 5 7 8 6 6 4 2 1
2 2 2 2 3 5 5 5 3 3 5 3 3 5 7 5 3 1 1 3 3 4 3 2 2 2 2 1 1 1
1 1 3 2 2 2 4 2 1
```

2021. május, idegennyelvű: Bányató

Informatika emelt szint

Azonosító jel:

4. Bányató

A bányató egy elhagyott külszíni bánya, amely egy idő után megtelik vízzel. Ebben a feladatban egy bányató mélységét kell elemeznie.

A tó felszínét sakktáblaszerűen 1 m oldalhosszúságú négyzetekre bontották, és minden ilyen négyzetben megmérték a tó mélységét. A mérést deciméter pontossággal végezték. A szárazföldet a 0 érték jelzi. A mérési adatokat egy téglalap alakú táblázatban rögzítették, például:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	3	0	0	0	0	0	8	10	10	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	19	10	0	11	16	18	10	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	9	17	35	5	11	11	22	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	19	75	64	53	61	25	0	10	4	11	17	17	0	27	61	13	0	0	0	0	0	0
6	0	2	0	18	75	58	64	25	49	0	0	34	74	73	67	20	55	75	3	15	2	10	0	0	0
7	0	18	25	23	71	32	34	62	24	0	0	43	55	58	51	28	75	58	2	22	35	9	3	10	0
8	0	6	12	35	76	47	51	62	0	0	0	32	65	40	84	51	40	40	47	65	0	0	26	21	0
9	0	6	29	40	76	43	73	88	0	0	0	33	88	76	73	38	56	28	4	86	80	0	15	16	0
10	0	0	14	31	24	42	52	63	42	27	0	50	50	42	67	84	51	53	33	46	0	0	7	3	0
11	0	13	29	27	80	34	38	57	68	72	26	55	22	81	76	60	34	51	4	13	0	0	0	0	0
12	0	12	26	27	65	33	77	49	73	38	89	35	80	36	76	77	88	79	18	13	18	10	0	0	0
13	0	12	25	37	59	65	20	57	33	48	84	75	48	33	34	92	86	97	89	93	82	86	3	11	0
14	0	0	0	35	53	59	38	37	69	36	54	52	55	36	36	96	81	92	95	98	82	90	0	0	0
15	0	0	6	56	62	47	47	60	58	25	20	38	64	47	29	40	44	23	25	0	0	0	0	0	0

Az ábrán az első oszlop, illetve az első sor a mérési adatok koordinátáját adja meg, például 12. sor 6. oszlopában lévő mérési eredmény 33 dm. (A tó medrét szürke háttér jelzi.)

Rendelkezésére áll a `melyseg.txt` nevű adatfájl, amelynek első két sorában az adatokat tartalmazó táblázat sorainak majd oszlopainak száma található. A fájlban ezt a mérési adatok követik soronként, az adatokat szóköz választja el egymástól. A fájlban a sorok és oszlopok azonosítói nem szerepelnek. Például egy 42 sorból és 25 oszlopból álló táblázat esetén az első 4 sor adatai a fájlban:

```
42
25
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 3 0 0 0 0 0 8 10 10 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 19 10 0 11 16 18 10 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 9 17 35 5 11 11 22 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0
…
```

Készítsen programot, amely az állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `banyato` néven! A megoldás során felhasználhatja, hogy a fájl legfeljebb 99 sort és legfeljebb 99 oszlopot tartalmaz. 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.

Informatika emelt szint | Azonosító jel: | | | | | | | | | | | | | | | |

1. Olvassa be és tárolja el a *melyseg.txt* állomány adatait, és annak felhasználásával oldja meg a következő feladatokat!
2. Kérje be egy mérési eredmény sor- és oszlopazonosítóját, majd írassa ki az adott helyen mért adatot a képernyőre! (A sorok és oszlopok számozása kezdődjön 1-gyel!)
3. Határozza meg a tó (vagyis az ábrán szürkével jelölt rész) felszínének területét, valamint a tó átlagos mélységét! Írassa ki a két eredményt a mintának megfelelően a képernyőre! A tó átlagos mélysége méterben kifejezve, két tizedesjegy pontossággal jelenjen meg!
4. Mekkora a tó legnagyobb mélysége, és hol a legmélyebb a tó? Jelenítse meg a választ a képernyőn! A legmélyebb pont koordinátáit a mintának megfelelően (*sor*; *oszlop*) formában írassa ki! Ha több ilyen mérési eredmény is van, mindegyik koordinátapárja jelenjen meg!
5. Milyen hosszú a tó partvonala, vagyis az ábrán a szürkével jelölt részt határoló vastag fekete vonal hossza? A partvonalhoz vegye hozzá a tóban lévő szigetek kerületét is! Írassa ki az eredményt a mintának megfelelően a képernyőre! (A megoldás során felhasználhatja, hogy a táblázat első és utolsó sorában és oszlopában minden adat 0.)
6. Kérje be a felhasználótól egy oszlop azonosítóját, és szemléltesse a *diagram.txt* szöveges állományban „sávdiagramon” a tó mélységét az adott oszlopban a következő módon! A sor elején jelenjen meg a mérési adat sorának azonosítója pontosan két számjeggyel, majd tegyen egymás mellé annyi csillagot (*), ahány méter az adott helyen a tó mélysége! A mérési adatokat a matematika szabályainak megfelelően kerekítse!

Példa a szöveges kimenetek kialakításához (a tizedesjel az alkalmazott fejlesztői környezettől függően eltérhet):

```
2. feladat
A mérés sorának azonosítója=12
A mérés oszlopának azonosítója=6
A mért mélység az adott helyen 33 dm
3. feladat
A tó felszíne: 646 m2, átlagos mélysége: 4,28 m
4. feladat
A tó legnagyobb mélysége: 98 dm
A legmélyebb helyek sor-oszlop koordinátái:
(14; 20)   (26; 11)   (32; 16)
5. feladat
A tó partvonala 270 m hosszú
6. feladat
A vizsgált szelvény oszlopának azonosítója=6
```

Példa a *diagram.txt* szöveges állomány tartalmára:

```
01
02
03**
04****
05******
06******
07***
08*****
…
```

45 pont

```
"""
2021. május idegennyelvű: Bányató
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("melyseg.txt")

sdb=int(betxt.readline().strip())
odb=int(betxt.readline().strip())

"""
A könnyebb feldolgozás kedvéért a sordbxoszlopdb-s táblázatot sorfolytonos listában tároljuk.
A transzformációk:
sor=index//odb +1, oszlop=index%odb +1
index=(sor-1)*odb+(oszlop-1)
"""

melysegek=[]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=list(map(int,darsor))
    melysegek+=adatsor

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

akts=int(input(f'A mérés sorának azonosítója (max. {sdb}) = ').strip())
akto=int(input(f'A mérés oszlopának azonosítója (max. {odb}) = ').strip())
print(f'A mért mélység az adott helyen {melysegek[(akts-1)*odb+akto-1]} dm.')



print("\n3. feladat\n")

to=list(filter(lambda m: m>0 ,melysegek))
print(f'A tó felszíne {len(to)} m^2, átlagos mélysége: {sum(to)/(10*len(to)):.2f} m.')



print("\n4. feladat\n")

maxm=max(to)
maxik=list(filter(lambda i: melysegek[i]==maxm ,range(len(melysegek))))
mhelyek=list(map(lambda i: f'({i//odb+1}, {i%odb+1})' ,maxik))
print(f'A tó legnagyobb mélysége: {maxm} dm.')
print("A legmélyebb helyek sor-oszlop koordinátái:")
print("\t".join(mhelyek))



print("\n5. feladat\n")

partvonal=0
for i,m in enumerate(melysegek):
    if m>0:
        szomszedok=[i-1,i+1,i-odb,i+odb] # tudjuk, hogy nincs a széleken
        partszomszedok=list(filter(lambda sz: melysegek[sz]==0 ,szomszedok))
        partvonal+=len(partszomszedok)

print(f'A tó partvonala {partvonal} m hosszú.')
```

```
print("\n6. feladat\n")

vo=int(input(f'A vizsgált szelvény oszlopának azonosítója (max. {odb}) = ').strip())

kitxt = open("diagram.txt","w")

for i in range(vo-1,len(melysegek),odb):
    kitxt.write(f'{i//odb+1:02}{round(melysegek[i]/10)*"*"}\n')


kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2021mid\EmeltInfo2021mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A mérés sorának azonosítója (max. 42) = 12
A mérés oszlopának azonosítója (max. 25) = 6
A mért mélység az adott helyen 33 dm.

3. feladat

A tó felszíne 646 m^2, átlagos mélysége: 4.28 m.

4. feladat

A tó legnagyobb mélysége: 98 dm.
A legmélyebb helyek sor-oszlop koordinátái:
(14, 20)(26, 11)(32, 16)

5. feladat

A tó partvonala 270 m hosszú.

6. feladat

A vizsgált szelvény oszlopának azonosítója (max. 25) = 6

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A diagram.txt szövegfile:

```
01                                                  22*********
02                                                  23********
03**                                                24*******
04****                                              25*******
05******                                            26*********
06******                                            27*******
07***                                               28*********
08*****                                             29****
09****                                              30****
10****                                              31*
11***                                               32****
12***                                               33***
13******                                            34*****
14******                                            35**
15*****                                             36
16**                                                37
17****                                              38****
18**                                                39****
19****                                              40**
20****                                              41*
21*******                                           42
```

2021. október: Sudoku

Informatika emelt szint | Azonosító jel:

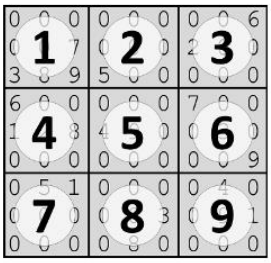
4. Sudoku

A sudoku egy logikai játék, melyben megadott szabályok szerint számjegyeket kell elhelyezni egy táblázatban. Ebben a feladatban 9×9-es táblázatot használunk.

A táblázat – az alábbi ábrának megfelelően – 9 darab 3×3-as résztáblázatra van felosztva. Minden résztáblázatot az 1, 2, 3, 4, 5, 6, 7, 8, 9 számokkal kell kitölteni úgy, hogy az egész 9×9-es táblázat minden sorában és minden oszlopában az 1...9 számok mindegyike pontosan egyszer forduljon elő. A rejtvény készítője előre ki szokta tölteni a táblázat bizonyos celláit. A rejtvényfejtő feladata kitölteni a maradék cellákat a leírt szabályoknak megfelelően.

A bemenetet tartalmazó szövegfájlok első 9 sorának mindegyike 9 egész számot tartalmaz, a játék kiindulási állapotának megfelelően. A kitöltetlen mezők helyén a 0 szám olvasható. A következő néhány sorban a játékos egy-egy lehetséges kitöltési lépését rögzítették. Egy lépést három egész szám ír le: a számot, amelyet a játékos be akar írni, majd a sor és az oszlop számát, ahova írni szeretné. A bemeneti fájl egy-egy sorában a számokat egy-egy szóköz választja el egymástól. A táblázat ellentmondásmentes, tehát megoldható feladatot ír le. A játékos által megtett lépések száma legalább 1, legfeljebb 10, közöttük lehet hibás.
Például:

```
0 0 0 0 0 0 0 0 6
0 0 7 0 0 0 2 0 0
3 8 9 5 0 0 0 0 0
6 0 0 0 0 0 7 0 0
1 2 8 4 9 0 0 3 0
0 0 0 0 0 0 0 0 9
0 5 1 0 0 0 0 4 0
0 0 0 0 0 3 0 9 1
0 0 0 0 8 0 0 0 0
9 2 4
1 2 1
5 9 9
7 2 2
```



A fenti példában a *nehez.txt* bemeneti fájl tartalma látható. A 10. sorban szereplő számok azt jelentik, hogy a 9-es értéket kell a 2. sor 4. helyére beírni. Az adott sorban és az adott oszlopban nem szerepel még a 9-es, sőt, az érintett négyzetben sem, így a lépéssel nem alakul ki hiba, megtehető. A 11. sorbeli lépés is megtehető. A 13. sor hibás lépést tartalmaz, mert a 2. sorban már szerepel a 7-es szám.

A jobb oldalon látható képen a körbe írt számok megadják, hogy az egyes 3×3 méretű résztáblákat milyen számmal azonosítjuk.

Készítsen programot, amely a bemeneti állományok egyikét felhasználva (*konnyu.txt*, *kozepes.txt*, *nehez.txt*) az alábbi kérdésekre válaszol! A program forráskódját mentse *sudoku* 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 egy fájl nevét, egy sor és egy oszlop sorszámát (1 és 9 közötti számot)! A későbbi feladatokat ezen értékek felhasználásával kell megoldania!

Informatika emelt szint | Azonosító jel: | | | | | | | | | | | | | | | |

2. Az előző feladatban beolvasott névnek megfelelő fájl tartalmát olvassa be, és tárolja el a táblázat adatait! Ha ezt nem tudja megtenni, akkor használja forrásként a rendelkezésre álló állományok egyikét!
3. Írja ki a képernyőre, hogy a beolvasott sor és oszlop értékének megfelelő hely…
   a. milyen értéket tartalmaz! Ha az adott helyen a 0 olvasható, akkor az „Az adott helyet még nem töltötték ki.” szöveget jelenítse meg!
   b. melyik résztáblázathoz tartozik!
4. Határozza meg a táblázat hány százaléka nincs még kitöltve! Az eredményt egy tizedesjegy pontossággal jelenítse meg a képernyőn!
5. Vizsgálja meg, hogy a fájlban szereplő lépések lehetségesek-e a beolvasott táblázaton! Tekintse mindegyiket úgy, mintha az lenne az egyetlen lépés az eredeti táblázaton, de ne hajtsa azt végre! Állapítsa meg, hogy okoz-e valamilyen ellentmondást a lépés végrehajtása! Írja ki a lépéshez tartozó három értéket, majd a következő sorba írja az alábbi megállapítások egyikét! Ha több megállapítás is igaz, elegendő csak egyet megjelenítenie.
   - „A helyet már kitöltötték”
   - „Az adott sorban már szerepel a szám”
   - „Az adott oszlopban már szerepel a szám”
   - „Az adott résztáblázatban már szerepel a szám”
   - „A lépés megtehető”

Minta a szöveges kimenetek kialakításához:

```
1. feladat
Adja meg a bemeneti fájl nevét! konnyu.txt
Adja meg egy sor számát! 1
Adja meg egy oszlop számát! 1

3. feladat
Az adott helyen szereplő szám: 5
A hely a(z) 1 résztáblázathoz tartozik.

4. feladat
Az üres helyek aránya: 17.3%

5. feladat
A kiválasztott sor: 2 oszlop: 4 a szám: 9
A helyet már kitöltötték.

A kiválasztott sor: 3 oszlop: 6 a szám: 7
A lépés megtehető.

A kiválasztott sor: 6 oszlop: 6 a szám: 3
A résztáblázatban már szerepel a szám.

A kiválasztott sor: 7 oszlop: 9 a szám: 8
Az adott oszlopban már szerepel a szám.
```

45 pont

```
"""
2021. október: Sudoku
@author Klemand66
"""

print("\n1. feladat\n")

fn=input("Adja meg a bemeneti fájl nevét (konnyu.txt, kozepes.txt vagy nehez.txt): ").strip()
akts=int(input("Adja meg egy sor számát (1 <= sor <= 9): ").strip())
akto=int(input("Adja meg egy oszlop számát (1 <= oszlop <= 9): ").strip())



print("\n2. feladat\n")

"""
A könnyebb feldolgozás kedvéért a 9x9-es táblázatot sorfolytonos listában tároljuk.
A transzformációk:
sor=index//9 +1, oszlop=index%9 +1
index=(sor-1)*9 +(oszlop-1)
"""

betxt = open(fn)

tabla, lepesek = [], []
for i in range(9):
    sor=betxt.readline()
    darsor=sor.strip().split()
    adatsor=list(map(int ,darsor))
    tabla+=(adatsor)

for sor in betxt:
    darsor=sor.strip().split()
    adatsor=list(map(int ,darsor))
    lepesek.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n3. feladat\n")

szam=tabla[(akts-1)*9 +(akto-1)]
if szam>0:
    print(f'Az adott helyen szereplő szám: {szam}')
else:
    print("Az adott helyet még nem töltötték ki.")
rt=((akts-1)//3)*3 + (akto-1)//3 +1
print(f'Az adott hely a(z) {rt}. résztáblázathoz tartozik.')



print("\n4. feladat\n")

uresdb=tabla.count(0)
print(f'Az üres helyek aránya: {100*uresdb/len(tabla):.1f}%.')
```

```
print("\n5. feladat")

for lepes in lepesek:
    si,oi,n = lepes[1], lepes[2], lepes[0]
    print(f'\nA kiválasztott sor: {si}, oszlop: {oi}, a szám: {n}.')
    megteheto=True
    akti=(si-1)*9 +(oi-1)
    if tabla[akti]>0:
        print("A helyet már kitöltötték.")
        continue
    aktsor=list(map(lambda i: tabla[i],range((si-1)*9,si*9)))
    if n in aktsor:
        print("Az adott sorban már szerepel a szám.")
        megteheto=False
    aktoszlop=list(map(lambda i: tabla[i],range(oi-1,oi-1+72,9)))
    if n in aktoszlop:
        print("Az adott oszlopban már szerepel a szám.")
        megteheto=False
    ki=((si-1)//3)*27 + ((oi-1)//3)*3
    aktrtabla=list(map(lambda i: tabla[i],[ki,ki+1,ki+2,ki+9,ki+10,ki+11,ki+18,ki+19,ki+20]))
    if n in aktrtabla:
        print("Az adott résztáblázatban már szerepel a szám.")
        megteheto=False
    if megteheto:
        print("A lépés megtehető.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2021okt\EmeltInfo2021okt.py ==================

1. feladat

Adja meg a bemeneti fájl nevét (konnyu.txt, kozepes.txt vagy nehez.txt): konnyu.txt
Adja meg egy sor számát (1 <= sor <= 9): 1
Adja meg egy oszlop számát (1 <= oszlop <= 9): 1

2. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

3. feladat

Az adott helyen szereplő szám: 5
Az adott hely a(z) 1. résztáblázathoz tartozik.

4. feladat

Az üres helyek aránya: 17.3%.

5. feladat

A kiválasztott sor: 2, oszlop: 4, a szám: 9.
A helyet már kitöltötték.

A kiválasztott sor: 3, oszlop: 6, a szám: 7.
A lépés megtehető.

A kiválasztott sor: 6, oszlop: 6, a szám: 3.
Az adott résztáblázatban már szerepel a szám.

A kiválasztott sor: 7, oszlop: 9, a szám: 8.
Az adott oszlopban már szerepel a szám.
Az adott résztáblázatban már szerepel a szám.

A befejezéshez nyomd meg az ENTER billentyűt!
```

2022. május: Építményadó

Informatika emelt szint | Azonosító jel: | | | | | | | | | | | | | | |

4. Építményadó

Egy Balaton-parti önkormányzat építményadót vezet be. Az adó mértéke a telken lévő építmény alapterületétől és a teleknek a Balatontól mért távolságától függ.

A telkeket a Balatonparttól mért távolságtól függően három sávba sorolták be. Az *A* sávba azok a telkek kerültek, amelyek 300 méternél közelebb vannak a tóhoz a *B* sáv az előzőn túl 600 méter távolságig terjed, a többi telek a *C* sávba tartozik. Az építmény után négyzetméterenként fizetendő összeg sávonként eltérő, azonban, ha az így kiszámított összeg nem éri el a 10.000 Ft-ot, akkor az adott építmény után nem kell adót fizetni.

A testületi döntést az Adó Ügyosztály egy mintával készítette elő, amely csupán néhány utca adatait tartalmazza. Ezek az adatok az `utca.txt` fájlban vannak. A fájl első sorában a három adósávhoz tartozó négyzetméterenként fizetendő összeg található *A, B, C* sorrendben, egy-egy szóközzel elválasztva:

```
800 600 100
…
33366 Aradi 8A C 180
22510 Aradi 8B C 137
90561 Aradi 10 C 168
…
```

A többi sorban egy-egy építmény adatai szerepelnek egy-egy szóközzel elválasztva. Az első a telek tulajdonosának ötjegyű adószáma; egy tulajdonosnak több telke is lehet. A második adat az utca neve, amely nem tartalmazhat szóközt. A harmadik adat a házszám, majd az adósáv megnevezése, végül az építmény alapterülete következik. A minta harmadik sorában például azt látjuk, hogy a *33366* adószámú tulajdonos telke az *Aradi* utca *8A*-ban található, és a *C* sávba eső telken álló építmény alapterülete *180* m^2.

A fájl legfeljebb 1000 telek adatait tartalmazza. A feladat megoldása során kihasználhatja, hogy a fájlban az adatok utca, azon belül pedig házszám szerinti sorrendben következnek.

Készítsen programot, amely az `utca.txt` állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `epitmenyado` 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 az `utca.txt` állományban talált adatokat, és annak felhasználásával oldja meg a következő feladatokat!
2. Hány telek adatai találhatók az állományban? Az eredményt írassa ki a mintának megfelelően a képernyőre!
3. Kérje be egy tulajdonos adószámát, és írassa ki a mintához hasonlóan, hogy melyik utcában, milyen házszám alatt van építménye! Ha a megadott azonosító nem szerepel az adatállományban, akkor írassa ki a ***„Nem szerepel az adatállományban.”*** hibaüzenetet!

Informatika emelt szint | Azonosító jel: | | | | | | | | | | | | | | | |

4. Készítsen függvényt `ado` néven, amely meghatározza egy adott építmény után fizetendő adót! A függvény paraméterlistájában szerepeljen az adósáv és az alapterület, visszaadott értéke pedig legyen a fizetendő adó! A következő feladatokban ezt a függvényt is felhasználhatja.
5. Határozza meg, hogy hány építmény esik az egyes adósávokba, és mennyi az adó összege adósávonként! Az eredményt a mintának megfelelően írassa ki a képernyőre!
6. Bár az utcák többé-kevésbé párhuzamosak a tó partjával, az egyes porták távolsága a parttól az utcában nem feltétlenül ugyanannyi. Emiatt néhány utcában – az ottani tulajdonosok felháborodására – egyes telkek eltérő sávba esnek. Listázza ki a képernyőre, hogy melyek azok az utcák, ahol a telkek sávokba sorolását emiatt felül kell vizsgálni! Feltételezheti, hogy minden utcában van legalább két telek.
7. Határozza meg a fizetendő adót tulajdonosonként! A tulajdonos adószámát és a fizetendő összeget írassa ki a mintának megfelelően a *fizetendo.txt* állományba! A fájlban minden tulajdonos adatai új sorban szerepeljenek, a tulajdonos adószámát egy szóközzel elválasztva kövesse az általa fizetendő adó teljes összege.

Példa a szöveges kimenetek kialakításához:

```
2. feladat. A mintában 543 telek szerepel.
3. feladat. Egy tulajdonos adószáma: 68396
Harmat utca 22
Szepesi utca 17
5. feladat
A sávba 165 telek esik, az adó 20805600 Ft.
B sávba 144 telek esik, az adó 13107000 Ft.
C sávba 234 telek esik, az adó 3479600 Ft.
6. feladat. A több sávba sorolt utcák:
Besztercei
Gyurgyalag
Icce
Kurta
Rezeda
Szepesi
```

Példa a *fizetendo.txt* fájl kialakításához:
(A fájl a megadott forrásállomány esetén 519 adatsort fog tartalmazni.)

```
38522 18000
86379 0
79906 12300
…
73850 204000
74143 100000
59801 563200
73011 70400
…
```

45 pont

```
"""
2022. május: Építményadó
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("utca.txt")

adosavok=["A","B","C"]
sor=betxt.readline()
darsor=sor.strip().split()
savadok=list(map(int,darsor))

telkek=[]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=[darsor[0],darsor[1],darsor[2],darsor[3],int(darsor[4])]
      #  0: adószám  1: utca  2: házszám  3: adósáv  4: alapterület
    telkek.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A mintában {len(telkek)} telek szerepel.')



print("\n3. feladat\n")

adsz=input("Kérem egy tulajdonos adószámát (pl. 68396): ").strip()
akttelkek=list(filter(lambda t: t[0]==adsz ,telkek))

if len(akttelkek)>0:
    for t in akttelkek:
        print(f'{t[1]} utca {t[2]}')
else:
    print("Nem szerepel az adatállományban.")



print("\n4. feladat\n")

def ado(adosav, alapterulet):
    aktindex=adosavok.index(adosav)
    ado=savadok[aktindex]*alapterulet
    if ado < 10000:
        return 0
    else:
        return ado

print(f'Az elkészített függvény:\n\n\
def ado(adosav, alapterulet):\n\
    aktindex=adosavok.index(adosav)\n\
    ado=savadok[aktindex]*alapterulet\n\
    if ado < 10000:\n\
        return 0\n\
    else:\n\
      return ado')



print("\n5. feladat\n")

for s in adosavok:
    aktsavtelkek=list(filter(lambda t: t[3]==s ,telkek))
    aktteruletek=list(map(lambda t: t[4] ,aktsavtelkek))
    aktadok=list(map(lambda a: ado(s,a) ,aktteruletek))
    print(f'{s} sávba {len(aktsavtelkek)} telek esik, az adó: {sum(aktadok):10,} Ft.')
```

```
print("\n6. feladat\n")

print("A több sávba sorolt utcák:")

utcalista, adosavlistak = [], []
for t in telkek:
    if t[1] in utcalista:
        aktindex=utcalista.index(t[1])
        if t[3] not in adosavlistak[aktindex]:
            adosavlistak[aktindex].append(t[3])
    else:
        utcalista.append(t[1])
        adosavlistak.append([t[3]])

for i,u in enumerate(utcalista):
    if len(adosavlistak[i])>1:
        print(u)



print("\n7. feladat\n")

kitxt = open("fizetendo.txt","w")

adoszamlista, adolista = [], []
for t in telkek:
    if t[0] in adoszamlista:
        aktindex=adoszamlista.index(t[0])
        adolista[aktindex]+=ado(t[3],t[4])
    else:
        adoszamlista.append(t[0])
        adolista.append(ado(t[3],t[4]))

for i,a in enumerate(adoszamlista):
    kitxt.write(f'{a} {adolista[i]}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2022maj\EmeltInfo2022maj.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A mintában 543 telek szerepel.

3. feladat

Kérem egy tulajdonos adószámát (pl. 68396): 68396
Harmat utca 22
Szepesi utca 17

4. feladat

Az elkészített függvény:

def ado(adosav, alapterulet):
    aktindex=adosavok.index(adosav)
    ado=savadok[aktindex]*alapterulet
    if ado < 10000:
        return 0
    else:
      return ado

5. feladat

A sávba 165 telek esik, az adó: 20,805,600 Ft.
B sávba 144 telek esik, az adó: 13,107,000 Ft.
C sávba 234 telek esik, az adó:  3,479,600 Ft.

6. feladat

A több sávba sorolt utcák:
Besztercei
Gyurgyalag
Icce
Kurta
Rezeda
Szepesi

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A fizetendo.txt szövegfile:

```
38522 18000
86379 0
79906 12300
92210 15200
49812 15400
26753 23100
97969 0
33366 18000
22510 13700
90561 16800
59966 29700
41456 24600
45156 28200
76032 12000
79028 21300
21210 27700
76500 15100
38722 14700
29556 13900
42220 0
66550 14700
90844 96000
78384 64800
29279 70400
18167 140800
72311 149600
15974 154400
82049 48000
76469 48000
29487 68000
26395 122400
39637 39200
97723 150400
84103 33600
19516 28800
65118 260600
74853 384000
76092 140000
86861 225600
98489 84000
15095 88000
31231 128800
34233 130200
39499 125400
53972 144000
50050 87000
21929 100800
87586 149400
47026 63000
58059 57600
43181 117600
17487 119400
85448 70800
42848 72000
50751 72000
24510 43200
58259 130800
17219 137400
23530 111600
51751 69600
30893 48000
...
14248 166400
26955 40000
78747 76000
26308 218400
74835 131200
61324 91800
56788 102600
26766 31200
57428 57600
93666 120600
87083 147600
75008 174000
30504 111600
98791 113400
33764 105000
74410 35400
19586 79800
98769 72000
47371 72000
23434 72000
38792 72000
88594 169200
63031 177000
90775 68400
34645 52800
66309 164400
15558 97200
31451 20900
73683 10000
23008 10000
56492 10000
30863 10000
47751 0
54115 16900
95826 10000
23231 10000
95645 0
25359 0
60493 12100
13779 0
75797 20200
86719 17300
88414 29900
83360 15400
16549 0
33724 0
33347 0
85862 104800
46616 81600
82966 58400
87011 114400
91002 212000
86409 238400
21252 64000
61735 162400
97812 67200
25820 78400
95126 37600
65000 61600
67529 123200
21824 238400
12345 107200
```

2022. május, idegennyelvű: Szakaszsebesség-ellenőrzés

Informatika emelt szint | Azonosító jel: | | | | | | | | | | | | | | |

4. Szakaszsebesség-ellenőrzés

A szakaszsebesség-ellenőrzést már több országban is alkalmazzák a közlekedés ellenőrzésére. Ennek lényege, hogy a járművek elhaladnak két egymástól több kilométerre lévő ellenőrzőkapunál. Mindkét ponton rögzítik a jármű rendszámát és az elhaladás időpontját. Majd a két időérték segítségével kiszámítják az átlagsebességet. Ha ez meghaladja az útszakaszon megengedett legnagyobb sebességet, akkor a jármű vezetője szabálysértést követ el. Rendelkezésünkre állnak egy 10 km-es kétszer egysávos főútvonal egyik sávjában rögzített szakaszsebesség mérésének adatai. Az adott szakasz zárt, azaz nincs felhajtási és lehajtási lehetőség.

A *meresek.txt* szövegállomány egy adott nap reggel 8 órától végzett 1 órányi mérés adatait tartalmazza. Minden egyes jármű csak egyszer szerepel a mérési adatok között. Egy jármű mérési adatai egy sorban szerepelnek egymástól szóközzel elválasztva. Egy sorban 9 adat szerepel, a jármű rendszáma (6 karakteren), a szakasz kezdeti- és végpontján rögzített időpont óra, perc, másodperc, ezredmásodperc formában. (A fájl olyan járművek adatait nem tartalmazza, amelyeknek a szakasz kezdeti- vagy végpontján nem volt mérési értéke.)

A sorok száma legfeljebb 1000. Az adatok a belépési ponton mért idő szerint rendezettek.
Például:

```
OXZ648 8 4 44 861 8 11 53 432
QUT385 8 4 53 376 8 9 28 185
QTS988 8 5 0 854 8 12 19 879
OTP604 8 5 2 263 8 12 21 288
```

A példában látható, hogy az QTS988 rendszámú jármű 8:5:0,854-kor haladt el a szakasz kezdetén lévő mérőnél és 8:12:19,879-kor a szakasz végén lévő mérőnél. Az átlagsebessége 82 km/h, ami a megtett út (10 km) és a megtételhez szükséges idő (0,1219 óra) hányadosa.

Készítsen programot, amely a *meresek.txt* állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *ellenorzes* 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. A tizedesszámok megjelenítésekor a használt programozási nyelvben alapértelmezett megjelenítési módot használja!

Az eredmény megjelenítését és a felhasználóval való kommunikációt a feladatot követő minta alapján valósítsa meg!

1. Olvassa be és tárolja el a *meresek.txt* állomány adatait!
2. Írja ki a képernyőre, hogy hány jármű adatait rögzítették a mérés során!
3. Határozza meg a rendelkezésre álló adatok segítségével, hogy 9 óra előtt hány jármű haladt át a szakasz végpontján! A kapott értéket írja ki a képernyőre!
4. Kérjen be a felhasználótól egy óra, perc értéket!
   a. Határozza meg, hogy abban a percben hány jármű haladt el a kezdő méréspontnál! Ha az adott percben nem haladt el jármű a méréspontnál, akkor a 0 értéket jelenítse meg!
   b. Számítsa ki a forgalomsűrűséget, amely a megadott időpontban kezdődő percben (pl.: ha a megadott óra perc 08:09 volt, akkor 08:09:00,000-08:09:59,999 között) az útszakaszon lévő járművek száma és az útszakasz hosszának hányadosa. Az értéket tizedes tört alakban jelenítse meg.

Informatika emelt szint Azonosító jel:

5. Mekkora volt a legnagyobb átlagsebességgel haladó járműnek a sebessége, és hány járművet hagyott le a mért szakasz végére? Amennyiben több legnagyobb átlagsebesség érték van, akkor elég az egyiket kiírnia. Az autó rendszámát, az átlagsebességet egész számként és a lehagyott járművek számát jelenítse meg!

6. Határozza meg, hogy a járművek hány százalékának az átlagsebessége haladta meg az útszakaszon megengedett legnagyobb sebességet (90 km/h)! Az értéket tizedes tört alakban jelenítse meg a minta szerint!

7. Készítsen egy szöveges állományt `buntetes.txt` néven, amely a gyorshajtók adatait tartalmazza! Ebbe a szöveges állományba azon járművek adatai kerüljenek be, amelyek átlagsebessége 104 km/h-nál nagyobb! A fájlban a jármű rendszáma, az átlagsebesség egész számként megjelenítve és a büntetés összege szerepeljen mértékegységgel, pontosvesszővel vagy tabulátorokkal elválasztottan! A büntetési tételeket a következő táblázat tartalmazza!

Átlagsebesség	Büntetés
104 km/h felett 121 km/h-ig	30 000 Ft
121 km/h felett 136 km/h-ig	45 000 Ft
136 km/h felett 151 km/h-ig	60 000 Ft
151 km/h felett	200 000 Ft

45 pont

Minta a szöveges kimenetek kialakításához:

```
2. feladat
A mérés során 687 jármű adatait rögzítették.

3. feladat
9 óra előtt 603 jármű haladt el a végponti mérőnél.

4. feladat
Adjon meg egy óra és perc értéket! 8 20
        a. A kezdeti méréspontnál elhaladt járművek száma: 12
        b. A forgalomsűrűség: 9,4

5. feladat
A legnagyobb sebességgel haladó jármű
        rendszáma: OKL564
        átlagsebessége: 137 km/h
        által lehagyott járművek száma: 33

6. feladat
A járművek 11,94%-a volt gyorshajtó.

A fájl elkészült.
```

A `buntetes.txt` fájl tartalma:

```
QZV314  128 km/h     45000 Ft
OJW811  107 km/h     30000 Ft
QUT385  131 km/h     45000 Ft
QRC440  128 km/h     45000 Ft
QJQ854  109 km/h     30000 Ft
...
```

```
"""
2022. május idegennyelvű: Szakaszsebesség-ellenőrzés
@author Klemand66
"""


print("\n1. feladat\n")

betxt = open("meresek.txt")

meresek=[]
for sor in betxt:
    d=sor.strip().split()
    rsz=d[0]
    belepes=int(d[1])*3600000+int(d[2])*60000+int(d[3])*1000+int(d[4])
    kilepes=int(d[5])*3600000+int(d[6])*60000+int(d[7])*1000+int(d[8])
    meresek.append([rsz,belepes,kilepes])

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A mérés során {len(meresek)} jármű adatait rögzítették.')



print("\n3. feladat\n")

aktki=9*3600000
elotteki=list(filter(lambda m: m[2]<aktki ,meresek))
print(f'9 óra előtt {len(elotteki)} jármű haladt el a végponti mérőnél.')


print("\n4. feladat\n")

aktperc=input("Adjon meg egy óra és perc értéket (Pl. 8 20): ")
d=aktperc.strip().split()
kezd=int(d[0])*3600000+int(d[1])*60000
vege=kezd+60000

kezdopontnal=list(filter(lambda m: m[1]>=kezd and m[1]<vege ,meresek))
print(f'\ta. A kezdeti mérőspontnál elhaladt járművek száma: {len(kezdopontnal)}.')

utszakaszon=list(filter(lambda m: m[2]>=kezd and m[1]<vege ,meresek))
print(f'\tb. A forgalomsűrűség: {len(utszakaszon)/10:.1f} jármű/km.')



print("\n5. feladat\n")

leggyorsabb=min(meresek, key=lambda m: m[2]-m[1])
maxseb=10*3600000/(leggyorsabb[2]-leggyorsabb[1])
lehagyottak=list(filter(lambda m: leggyorsabb[1]>m[1] and leggyorsabb[2]<m[2] ,meresek))

print("A legnagyobb sebességgel haladó jármű")
print(f'\trendszáma: {leggyorsabb[0]}')
print(f'\tátlagsebessége: {int(maxseb)} km/h')
print(f'\táltal lehagyott járművek száma: {len(lehagyottak)}')



print("\n6. feladat\n")

minido=3600000*10/90
gyorshajtok=list(filter(lambda m: m[2]-m[1]<minido ,meresek))
print(f'A járművek {100*len(gyorshajtok)/len(meresek):.2f}%-a volt gyorshajtó.')
```

```
print("\n7. feladat\n")

def kiszabo(be,ki):
    seb=int(10*3600000/(ki-be))
    if seb>151:
        return [seb,200000]
    elif seb>136:
        return [seb,60000]
    elif seb>121:
        return [seb,45000]
    elif seb>104:
        return [seb,30000]
    else:
        return 0

kitxt = open("buntetes.txt","w")

for gy in gyorshajtok:
    b=kiszabo(gy[1],gy[2])
    if b!=0:
        kitxt.write(f'{gy[0]}  {b[0]} km/h\t{b[1]} Ft\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2022mid\EmeltInfo2022mid.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A mérés során 687 jármű adatait rögzítették.

3. feladat

9 óra előtt 603 jármű haladt el a végponti mérőnél.

4. feladat

Adjon meg egy óra és perc értéket (Pl. 8 20): 8 20
        a. A kezdeti méréspontnál elhaladt járművek száma: 12.
        b. A forgalomsűrűség: 9.4 jármű/km.

5. feladat

A legnagyobb sebességgel haladó jármű
        rendszáma: OKL564
        átlagsebessége: 137 km/h
        által lehagyott járművek száma: 33

6. feladat

A járművek 11.94%-a volt gyorshajtó.

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A buntetes.txt szövegfile:

```
QZV314  128 km/h45000 Ft                OSI922  132 km/h45000 Ft
OJW811  107 km/h30000 Ft                QNG315  122 km/h45000 Ft
QUT385  131 km/h45000 Ft                OWC656  135 km/h45000 Ft
QRC440  128 km/h45000 Ft                OPC814  127 km/h45000 Ft
QJQ854  109 km/h30000 Ft                QXW101  117 km/h30000 Ft
QMR120  121 km/h30000 Ft                OSY953  136 km/h45000 Ft
OKL564  137 km/h60000 Ft                OJY688  117 km/h30000 Ft
QJY312  120 km/h30000 Ft                QTP774  133 km/h45000 Ft
OZB517  119 km/h30000 Ft                QJG162  111 km/h30000 Ft
OZO186  113 km/h30000 Ft                QEO586  128 km/h45000 Ft
QKQ540  133 km/h45000 Ft                OKY535  110 km/h30000 Ft
OJZ188  134 km/h45000 Ft                OIP865  135 km/h45000 Ft
ODB512  105 km/h30000 Ft                QLO627  134 km/h45000 Ft
OLY209  126 km/h45000 Ft                OPR547  105 km/h30000 Ft
QAJ541  109 km/h30000 Ft                ONT392  126 km/h45000 Ft
QJL973  126 km/h45000 Ft
```

2022. október: Jeladó

Informatika emelt szint | Azonosító jel:

4. Jeladó

Az állatok mozgását ma már rutinszerűen figyelik a rájuk rögzített jeladók segítségével. Ebben a feladatban egy ilyen jeladó által továbbított adatokat kell feldolgoznia.

Az itt használt jeladó úgy működik, hogy helyének ***x*** és ***y*** koordinátáját továbbítja. Jelet küld, ha a legutolsó küldés óta bármely koordináta változása elérte a 10 egységet. Ha nem történt ekkora elmozdulás, 5 perc elteltével akkor is mindenképpen jelenti helyét. A vevőegység egy fájlban rögzíti a jel érkezési idejét és a pozíciót. Előfordulhat, hogy a vétel meghiúsul, ezért lehetnek egymást követő adatsorok, amelyek között 5 percnél több idő telik el, vagy a koordináták változása 10 egységnél nagyobb.

Rendelkezésére áll a `jel.txt` nevű adatfájl, amely egy napról tartalmaz adatokat időrendben. Soraiban öt egész szám található, egymástól egy-egy szóközzel elválasztva. Az első három szám a jeladás időpontját (óra, perc, másodperc) adja meg, a negyedik szám az x, az ötödik az y koordináta. A sorok száma legfeljebb 1000, a koordináták -10 000 és 10 000 közötti értékek lehetnek.

Például:

```
…
3 21 19 126 639
3 26 19 131 641
3 27 55 124 651
3 31 50 134 649
…
4 19 11 126 42
4 29 11 128 36
4 32 21 130 7
…
```

A példa első csoportjában a második sor megmutatja, hogy a jeladó 5 egységnyit mozdult x, 2 egységnyit pedig y irányban 5 perc alatt. A harmadik bejegyzés azért született, mert y irányban 10 egységnyit mozdult el a jeladó, a negyedik bejegyzés pedig egy x irányú 10 egységnyi elmozdulást jelez.

A példa második csoportjában a második sor adataiból látszik, hogy legalább egyszer nem jutott el a jel a vevőhöz, mert 5 percnél több idő telt el az előző vételtől, de az eltelt idő a 10 percet nem haladja meg. A második és harmadik vétel által jelzett pozíciók y koordinátája 29 egységnyivel eltér, ezért legalább két vétel nem került rögzítésre.

Készítsen programot, amely az állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `jelado` 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: `5. 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 a `jel.txt` állomány tartalmát, tárolja el a rögzített jelek adatait, és azok felhasználásával oldja meg a következő feladatokat!
2. Kérje be a felhasználótól egy jel sorszámát (a sorszámozás 1-től indul), és írja a képernyőre az adott jeladáshoz tartozó ***x*** és ***y*** koordinátát!

Informatika emelt szint | Azonosító jel: | | | | | | | | | | | | | | | |

3. Készítsen függvényt `eltelt` néven, amely megadja, hogy a paraméterként átadott két időpont között hány másodperc telik el! A két időpontot, mint paramétert tetszőleges módon átadhatja. Használhat három-három számértéket, két tömböt vagy listát, de más, a célnak megfelelő típusú változót is. Ezt a függvényt később használja fel legalább egy feladat megoldása során!
4. Adja meg, mennyi idő telt el az első és az utolsó észlelés között! Az időt *óra:perc:mperc* alakban írja a képernyőre!
5. Határozza meg azt a legkisebb, a koordináta-rendszer tengelyeivel párhuzamos oldalú téglalapot, amelyből nem lépett ki a jeladó! Adja meg a téglalap bal alsó és jobb felső sarkának koordinátáit!
6. Írja a képernyőre, hogy mennyi volt a jeladó elmozdulásainak összege! Úgy tekintjük, hogy a jeladó két pozíciója közötti elmozdulása a pozíciókat összekötő egyenes mentén történt. Az összeget három tizedes pontossággal jelenítse meg! A kiírásnál a tizedesvessző és tizedespont kiírása is elfogadott. Az i-edik és az i+1-edik pontok távolságát a $\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$ képlet segítségével határozhatja meg.
7. Írja a `kimaradt.txt` fájlba a kimaradt észlelésekkel kapcsolatos adatokat! A kimeneti fájlban azok a bemeneti állományban rögzített vételi időpontok jelenjenek meg, amelyek előtt közvetlenül egy vagy több észlelés kimaradt! Az időpont mellett – a mintának megfelelően – jelenjen meg, hogy legalább hány jel maradt ki, és az is, hogy miből következtet a hiányra! Ha idő- és koordináta-eltérésből is adódik jelkimaradás, akkor a nagyobb értéket írja ki! Ha az időeltérés és a koordináták eltérése alapján is ugyanannyi jelkimaradásra következtetünk, akkor bármelyiket kiírhatja.

Példa a szöveges kimenetek kialakításához:

```
2. feladat
Adja meg a jel sorszámát! 3
x=126 y=639

4. feladat
Időtartam: 18:52:40

5. feladat
Bal alsó: 4 639, jobb felső: 147 727

6. feladat
Elmozdulás: 2007.677 egység
```

Minta a `kimaradt.txt` fájl tartalmára

```
4 25 33 időeltérés 1
4 55 33 koordináta-eltérés 1
5 5 33 időeltérés 1
6 22 42 időeltérés 2
6 32 42 koordináta-eltérés 2
```

45 pont

```
"""
2022. október: Jeladó
@author Klemand66
"""

from math import *


print("\n1. feladat\n")

betxt = open("jel.txt")

jelek=[]
for sor in betxt:
    d=sor.strip().split()
    t=int(d[0])*3600+int(d[1])*60+int(d[2])
    x, y = int(d[3]), int(d[4])
    jelek.append([t,x,y])

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

ssz=int(input(f'Adja meg a jel sorszámát (1 <= sorszám <= {len(jelek)}): ').strip())
print(f'x={jelek[ssz-1][1]} y={jelek[ssz-1][2]}')



print("\n3. feladat\n")

def eltelt(t1,t2):
    return t2-t1

print(f'Az elkészített függvény:\n\n\
def eltelt(t1,t2):\n\
    return t2-t1')



print("\n4. feladat\n")

def konvertalo(t):
    mp=str(t%60)
    t=t//60
    perc=str(t%60)
    ora=str(t//60)
    return [ora,perc,mp]

print(f'Az időtartam: {":".join(konvertalo(eltelt(jelek[0][0],jelek[-1][0])))}')



print("\n5. feladat\n")

xmin=min(jelek, key=lambda j: j[1])[1]
# A min() függvény a listát adja meg, annak a megfelelő eleme kell.
xmax=max(jelek, key=lambda j: j[1])[1]
ymin=min(jelek, key=lambda j: j[2])[2]
ymax=max(jelek, key=lambda j: j[2])[2]

print(f'Bal alsó: {xmin} {ymin}, jobb felső: {xmax} {ymax}')



print("\n6. feladat\n")

elmozdulasok=list(map(lambda i: \
    sqrt((jelek[i][1]-jelek[i+1][1])**2 + (jelek[i][2]-jelek[i+1][2])**2), range(len(jelek)-1)))
print(f'Elmozdulás: {sum(elmozdulasok):.3f} egység')
```

```
print("\n7. feladat\n")

kitxt = open("kimaradt.txt","w")

for i in range(1,len(jelek)):
    idokul=eltelt(jelek[i-1][0],jelek[i][0])
    idokim=idokul//300 -1
    if idokul%300>0:
        idokim+=1
    koordkul=max(abs(jelek[i][1]-jelek[i-1][1]),abs(jelek[i][2]-jelek[i-1][2]))
    koordkim=koordkul//10 -1
    if koordkul%10>0:
        koordkim+=1
    if idokim + koordkim >0:
        aktido=" ".join(konvertalo(jelek[i][0]))
        if idokim>=koordkim:
            kitxt.write(f'{aktido} időeltérés {idokim}\n')
        else:
            kitxt.write(f'{aktido} koordináta-eltérés {koordkim}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltInfo2022okt\EmeltInfo2022okt.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Adja meg a jel sorszámát (1 <= sorszám <= 250): 3
x=126 y=639

3. feladat

Az elkészített függvény:

def eltelt(t1,t2):
    return t2-t1

4. feladat

Az időtartam: 18:52:40

5. feladat

Bal alsó: 4 639, jobb felső: 147 727

6. feladat

Elmozdulás: 2007.677 egység

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A kimaradt.txt szövegfile:

```
4 25 33 időeltérés 1
4 55 33 koordináta-eltérés 1
5 5 33 időeltérés 1
6 22 42 időeltérés 2
6 32 42 koordináta-eltérés 2
```

Digitális kultúra mintafeladatsor: Kongresszus

4. Kongresszus[2] *(50 pont)*

A Radiológia nemzetközi napjához kapcsolódóan négynapos nemzetközi radiológiai kongresszust szervez egy szakmai bizottság november 5-e és 8-a között, melyen több neves előadó tart majd egymást követő előadásokat. A program minden nap 8 órakor kezdődik, és a tervek szerint minden előadás után 20 perces kötetlen beszélgetésre, vitára adnak lehetőséget az előadók az adott témáról. Azokon a napokon, amikor a program a délutánba is belenyúlik, a délben vagy utána véget érő első előadást és vitát egyórás ebédszünet követi majd.

Az `eloadasok.txt` tabulátorokkal (→) tagolt állomány sorai tartalmazzák az előadások adatait a következőképpen:

```
Carlo Catalano → 11 → 5 → 5 → 33 → Radiology and COVID-19 → -
Kis Ida → 11 → 6 → 5 → 36 → Sport és képalkotás → projektor
Kis Ida → 11 → 5 → 7 → 37 → Betegbiztonság → sötétítés, mikrofon
Minerva Becker → 11 → 7 → 2 → 36 → COVID-19 Database → -
Széll Péter András → 11 → 6 → 4 → 34 → Neuroradiológia → notebook
```

Elől az előadó neve szerepel, majd az előadás hónapja (ez mindig 11) és napja (5, 6, 7 vagy 8) következik. A harmadik szám az előadás napon belüli sorszáma, míg a következő az előadás tervezett hossza percben megadva. Ezt követi az előadás címe, majd opcionálisan az előadáshoz biztosítandó eszközök felsorolása vagy kötőjel, ha nem igényel eszközöket. A fenti minta harmadik sora szerint például Kis Ida november 5-én (1. előadási nap) 7. előadóként 37 perces „Betegbiztonság” című előadással készül, amihez elsötétíthető termet és mikrofont kell a szervezőknek biztosítania.

Készítsen `kongresszus` néven programot 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 az eloadasok.txt szöveges állomány adatait! Amennyiben nem tudja elvégezni a beolvasást, billentyűzetről vigye be az első nap adatait, és a továbbiakban ezekkel dolgozzon!

2. Az előadások a fájlban az előadó neve, azon belül pedig az előadás címe szerint lettek rendezve, de a szervezőknek hasznosabb lenne egy időrend szerinti lista. Írja ki a képernyőre az előadók nevét naponkénti bontásban az előadások sorrendjében! Az eredményt a következő formában jelenítse meg:

```
november 5.:
  1. Kovács Lajos Tamás: Megnyitó
  2. Sugár Zóra: Sport és képalkotás?
…
november 6.:
  1. Bánkövi Alajos: Csernobil és az orvosi radiológia
…
```

3. Számítsa ki, hogy mennyi lesz naponta az előadások vita nélküli összes ideje! Az eredményt óra:perc formában adja meg!
   Például: `2. nap: 5:25`

[2] Forrás: a feladat Bíró Zsolt „10 próbaérettségi informatikából” című könyve egyik feladatának átdolgozása.

4. Ki tartja majd a tervek szerint a leghosszabb előadást november 6-án? Írja ki az előadó nevét és az előadás hosszát percben! Több előadó esetén mindegyik nevét jelenítse meg!

5. Mikor érnek véget az egyes napokon az előadások? Írja ki az időpontokat a képernyőre!
Például: `november 5.: 18:19`

6. Mikor kezdődik az ebédszünet a harmadik napon? Írjon egész mondatos választ a képernyőre!

7. Vizsgálja meg, hogy jelentkeztek-e azonos nevű előadók! Ha igen, akkor a neveket és az előfordulások számát írja ki a képernyőre, ellenkező esetben a *„Nem találtam egyező neveket.”* szöveg jelenjen meg!
Például:
```
Kovács Lajos 3
Szabó Géza 2
```

8. Kérjen be billentyűzetről egy napot (5-8) és egy napon belüli tetszőleges időpontot (óra [0-23], perc [0-59])! Írja ki a képernyőre, hogy milyen esemény lesz éppen a megadott időben! A lehetséges válaszok: *„Előadás”*, *„Vita”*, *„Ebédszünet”*, *„Már véget ért”*, *„Még nem kezdődött el”*.

9. A kongresszus dokumentálásához szükség van a pontos időrend elkészítésére. Írja az `idorend.txt` fájlba (ha nem tud fájlba írni, akkor a képernyőre) a kongresszus adatait a következőképpen:

```
november 5.
8:00-8:32 Kovács Lajos Tamás: Megnyitó (mikrofon, lézermutató)
8:32-8:52 Vita
8:52-9:35 Sugár Zóra: Sport és képalkotás? (-)
…
11:43-12:16 Carlo Catalano: International Day of Radiology (-)
12:16-12:36 Vita
12:36-13:36 Ebéd
13:36-14:17 Madocsai Gáspár: Cardiovascularis képalkotás (-)
…
```

Ügyeljen az időpontok helyes formázására!

```
"""
2022. digkult minta: Kongresszus
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("eloadasok.txt", encoding="utf-8")
# Ügyelni kell az ékezetes forrásfájl kódolására.

ho, eloadasok = "november", []
for sor in betxt:
    d=sor.strip().split("\t")
    nev=d[0] # d[1] mindenütt 11, külön tároljuk a hónap nevét
    nap=int(d[2])
    ssz, hossz, cim, kell, = int(d[3]), int(d[4]), d[5], d[6]
    eloadasok.append([nev, nap, ssz, hossz, cim, kell])
                  #    0    1    2     3     4     5
betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

eloadasok.sort(key=lambda e: e[2]) # először a mellékszempont szerint rendezünk
eloadasok.sort(key=lambda e: e[1])

print("  Az előadások időrendben:\n")
for n in range(5,9):
    print(f'  {ho} {n}.:')
    akteloadasok=list(filter(lambda e: e[1]==n ,eloadasok))
    for a in akteloadasok:
        print(f'\t{a[2]}. {a[0]}: {a[4]}')
    print()



print("3. feladat\n")

def konvertalo(t):
    perc=t%60
    ora=t//60
    return f'{ora}:{perc:02}'

print("  Az előadások vita nélküli összideje:")
for n in range(1,5):
    akteloadasok=list(filter(lambda e: e[1]==n+4 ,eloadasok))
    hosszak=list(map(lambda i: akteloadasok[i][3] ,range(len(akteloadasok))))
    print(f'\t{n}. nap: {konvertalo(sum(hosszak))}')



print("\n4. feladat\n")

akteloadasok=list(filter(lambda e: e[1]==6 ,eloadasok))
leghosszabb=max(akteloadasok, key=lambda e: e[3])
maxhossz=leghosszabb[3]
print(f'  November 6-án a leghosszabb előadás(ok) hossza: {maxhossz} perc.')
leghosszabbak=list(filter(lambda a: a[3]==maxhossz ,akteloadasok))
eloadok=list(map(lambda l: l[0] ,leghosszabbak))
print(f'\tAz előadó(k): {", ".join(eloadok)}')



print("\n5. feladat\n")

print("  Az előadások vége:")
for n in range(5,9):
    akteloadasok=list(filter(lambda e: e[1]==n ,eloadasok))
    hosszak=list(map(lambda i: akteloadasok[i][3] ,range(len(akteloadasok))))
    vege=8*60+sum(hosszak)+len(hosszak)*20
    if vege>12*60:
        vege+=60
    print(f'\tnovember {n}.: {konvertalo(vege)}')
```

```
print("\n6. feladat\n")

akteloadasok=list(filter(lambda e: e[1]==7 ,eloadasok))
ebedido=8*60
for a in akteloadasok:
    ebedido+=a[3]+20
    if ebedido>=12*60:
        print(f'  A harmadik napon {konvertalo(ebedido)}-kor kezdődik az ebédszünet.')
        break



print("\n7. feladat\n")

nevlista, nevdblista = [], []
for e in eloadasok:
    if e[0] in nevlista:
        aktindex=nevlista.index(e[0])
        nevdblista[aktindex]+=1
    else:
        nevlista.append(e[0])
        nevdblista.append(1)

if max(nevdblista)<2:
    print("  Nem találtam egyező neveket.")
else:
    print("  A következő egyező neveket találtam:")
    for i,nev in enumerate(nevlista):
        if nevdblista[i]>1:
            print(f'\t{nev} {nevdblista[i]}')



print("\n8. feladat\n")

aktnap=int(input(f'Kérem a kongresszus egyik napját (5 <= nap <= 8): ').strip())
sor=input(f'Kérek egy időpontot óra perc alakban (0<= óra <= 23 és 0<= perc <=59, pl. 10 56): ')
darsor=sor.strip().split()
aktido=int(darsor[0])*60 + int(darsor[1])

akteloadasok=list(filter(lambda e: e[1]==aktnap ,eloadasok))
kezd=8*60
programok=["Még nem kezdődött el.", "Előadás", "Vita", "Ebédszünet", "Már véget ért."]
programsavok=[[range(kezd)], [], [], [], []]

for a in akteloadasok:
    vege=kezd+a[3]
    programsavok[1].append(range(kezd,vege))
    programsavok[2].append(range(vege,vege+20))
    kezd=vege+20
    if kezd>=12*60 and len(programsavok[3])==0:
        programsavok[3].append(range(kezd,kezd+60))
        kezd=kezd+60
programsavok[4].append(range(kezd,24*60))

for i,ps in enumerate(programsavok):
    for j in range(len(ps)):
        if aktido in ps[j]:
            aktindex=i
            print(f'  {programok[i]}')
```

```
print("\n9. feladat\n")

kitxt = open("idorend.txt","w")

for n in range(5,9):
    kitxt.write(f'  {ho} {n}.\n')
    akteloadasok=list(filter(lambda e: e[1]==n ,eloadasok))
    kezd, ebed = 8*60, False
    for a in akteloadasok:
        vege=kezd+a[3]
        kitxt.write(f'  {konvertalo(kezd)}-{konvertalo(vege)} {a[0]}: {a[4]} ({a[5]})\n')
        kitxt.write(f'  {konvertalo(vege)}-{konvertalo(vege+20)} Vita\n')
        kezd=vege+20
        if kezd>=12*60 and not ebed:
            kitxt.write(f'  {konvertalo(kezd)}-{konvertalo(kezd+60)} Ebéd\n')
            ebed=True
            kezd=kezd+60

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================== RESTART: C:\Python\KlEmProgPy\EmeltDigKultminta\EmeltDigKultminta.py ==================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

  Az előadások időrendben:

  november 5.:
        1. Kovács Lajos Tamás: Megnyitó
        2. Sugár Zóra: Sport és képalkotás?
        3. Fejes Anikó: Onkológia és képalkotás
        4. Bánkövi Alajos: Mesterséges intelligencia
        5. Carlo Catalano: International Day of Radiology
        6. Madocsai Gáspár: Cardiovascularis képalkotás
        7. Kis Ida: Betegbiztonság
        8. Dobos Ildikó: Neuroradiológia a gyakorlatban
        9. Kaján Vidor: Betegbiztonság (szakdolgozói szekció)
        10. Pesti János: Intervenciós radiológia

  november 6.:
        1. Bánkövi Alajos: Csernobil és az orvosi radiológia
        2. Kövesdi Márton: Radiológusok és a halálozási kockázat
        3. Schmidt Benedek: Kontasztanyagok
        4. Széll Péter András: Neuroradiológia
        5. Kis Ida: Sport és képalkotás
        6. Fekete Emese: Mellkas és kardiovaszkuláris radiológia
        7. Palkó András: A hasi parenchymás szervek
        8. Lelkes Péter: A Magyar Orvosi Kamara Radiológiai tagozata
        9. Kerekes Erika: Gyermekradiológia

  november 7.:
        1. Carlo Catalano: Radiology and COVID-19
        2. Minerva Becker: COVID-19 Open Radiology Database
        3. Imre Géza: Sürgősségi képalkotás a COVID tükrében
        4. Porkoláb Bernadett: COVID-19 képalkotás
        5. Katrine Riklund: European Congress of Radiology (ECR)
        6. Csete János: A COVID-19 egészségügyi áldozatai
        7. Carlo Catalano: The impact of COVID-19 on radiology trainees
        8. Görbe Ferenc: A gyomor-bélrendszer radiológiája

  november 8.:
        1. Palkó András: Szakmai minőségbiztosítás
        2. Kovács Lajos Tamás: Oktatás, képzés
        3. Palkó András: Záró előadás

3. feladat

  Az előadások vita nélküli összideje:
        1. nap: 5:59
        2. nap: 5:29
        3. nap: 5:21
        4. nap: 1:30

4. feladat

  November 6-án a leghosszabb előadás(ok) hossza: 44 perc.
        Az előadó(k): Palkó András, Kerekes Erika
```

```
5. feladat

  Az előadások vége:
        november 5.: 18:19
        november 6.: 17:29
        november 7.: 17:01
        november 8.: 10:30

6. feladat

  A harmadik napon 12:46-kor kezdődik az ebédszünet.

7. feladat

  A következő egyező neveket találtam:
        Kovács Lajos Tamás 2
        Bánkövi Alajos 2
        Carlo Catalano 3
        Kis Ida 2
        Palkó András 3

8. feladat

Kérem a kongresszus egyik napját (5 <= nap <= 8): 7
Kérek egy időpontot óra perc alakban (0<= óra <= 23 és 0<= perc <=59, pl. 10 56): 12 45
  Vita

9. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!


Az idorend.txt szövegfile:

  november 5.
  8:00-8:32 Kovács Lajos Tamás: Megnyitó (mikrofon, lézermutató)
  8:32-8:52 Vita
  8:52-9:35 Sugár Zóra: Sport és képalkotás? (-)
  9:35-9:55 Vita
  9:55-10:23 Fejes Anikó: Onkológia és képalkotás (-)
  10:23-10:43 Vita
  10:43-11:23 Bánkövi Alajos: Mesterséges intelligencia (-)
  11:23-11:43 Vita
  11:43-12:16 Carlo Catalano: International Day of Radiology (-)
  12:16-12:36 Vita
  12:36-13:36 Ebéd
  13:36-14:17 Madocsai Gáspár: Cardiovascularis képalkotás (-)
  14:17-14:37 Vita
  14:37-15:14 Kis Ida: Betegbiztonság (sötétítés, mikrofon)
  15:14-15:34 Vita
  15:34-16:22 Dobos Ildikó: Neuroradiológia a gyakorlatban (-)
  16:22-16:42 Vita
  16:42-17:09 Kaján Vidor: Betegbiztonság (szakdolgozói szekció) (-)
  17:09-17:29 Vita
  17:29-17:59 Pesti János: Intervenciós radiológia (-)
  17:59-18:19 Vita
  november 6.
  8:00-8:29 Bánkövi Alajos: Csernobil és az orvosi radiológia (-)
  8:29-8:49 Vita
  ...
```

2022. október, digitális kultúra: Virágágyások

Digitális kultúra
emelt szint

Azonosító jel:

3. Virágágyások

Egy frissen épült óriási lakóparkba költözők elhatározták, hogy a még kopár környezetet megszépítik. A lakóparkot határoló kerítés mellett kijelölték a virágágyások helyét, és sorszámokkal látták el azokat. A sorszámozást, amely eggyel kezdődik, az egyetlen bejárat jobb oldalán lévő ágyással kezdték, és a bejárat bal oldalán lévő ágyással fejezték be. A lakók megadhatták, hogy mely szomszédos ágyásokat szeretnék beültetni. A felajánlásban megadták azt is, hogy milyen színű virággal.

Minden ágyásnak, az első és az utolsó kivételével, két szomszédja van, az eggyel kisebb és az eggyel nagyobb sorszámú, valamint az utolsó az első szomszédjának számít.

A `felajanlas.txt` állomány első sorában a virágágyások száma szerepel, amely nem haladja meg a 3000 értéket. A fájl további soraiban két szám és egy betű található, egymástól egy-egy szóközzel elválasztva, amely egy lakó felajánlását írja le. (A felajánlások száma nem haladja meg a virágágyások számát.) Az első szám az első, a második az utolsó beültetni kívánt ágyás sorszáma. A betű az angol ábécé nagybetűje, amely a választott színt jelöli. Ha az első szám nagyobb, mint a második, akkor a választott intervallumba a sorszám szerinti utolsó és első ágyás, tehát a bejárat két oldala is beletartozik. Mindenki csak egy felajánlást tehet, de egy ágyás több felajánlásban is szerepelhet, azaz a felajánlásokban szereplő intervallumok átfedhetik egymást. Az ültetést a felajánlások sorrendjében végzik el a lakók.

Például:

```
100
91 98 K
95 5 Z
63 76 N
…
```

A fenti példában az ágyások száma 100. A sorban az első felajánló kezdi az ültetést, 8 ágyást ültet be, a második felajánló 11 ágyás beültetésére vállalkozott, amelyből négyet már az első felajánló beültet, ezért csak 7 ágyást ültet be. A második felajánló ülteti be a bejárat két oldalán lévő 1. és 100. virágágyást is.

Készítsen programot, amely az állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse `park` 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: `5. 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 a `felajanlas.txt` állomány tartalmát, tárolja el a felajánlások adatait, és annak felhasználásával oldja meg a következő feladatokat!
2. Írja ki, hány felajánlást tartalmaz az állomány!
3. Jelenítse meg a képernyőn azon felajánlások sorszámát, amelyek a bejárat bal és jobb oldalán található ágyást is beültetnék! A sorszámokat egy-egy szóközzel válassza el egymástól!

A feladat a következő oldalon folytatódik.

Digitális kultúra emelt szint | Azonosító jel: | | | | | | | | | | | | | | | |

4. Kérje be a felhasználótól egy ágyás sorszámát! A tesztelés során használhatja az 1. ágyást, amelyet többen is beültetnének és a 269. ágyást, amelynek beültetésére senki sem vállalkozik.

    a) Írja a képernyőre, hogy hány felajánlásban szerepel ez az ágyás!

    b) Adja meg, milyen színű lesz ez az ágyás, ha mindenki a felajánlások sorrendjében végzi el az ültetést, de nem ültet, ha másvalaki előtte már ültetett oda! Ha nem ültetett oda senki, akkor „Ezt az ágyást nem ültetik be.” szöveget jelenítse meg!

    c) Adja meg, milyen színekben pompázna ez az ágyás, ha az eredeti tervvel ellentétesen minden felajánló elültetné virágait! Minden színt csak egyszer tüntessen fel! Az egyes színeket szóközökkel válassza el egymástól! Ha nem ültettek oda virágot, ne jelenítsen meg semmit!

5. A felajánlások alapján több eset lehetséges. Határozza meg, melyik teljesül! Az idézőjelek közötti szöveget írja a képernyőre!

    - „Minden ágyás beültetésére van jelentkező.”
    - Ha nincs minden ágyásra jelentkező, de a felajánlásokban vállalt ágyások számának összege nagyobb vagy egyenlő, mint, az ágyások száma, akkor: „Átszervezéssel megoldható a beültetés.”
    - Ha kevesebb ágyás beültetésére vállalkoztak, mint amennyi van: „A beültetés nem oldható meg.”

6. A beültetést a felajánlások sorrendjében végezték el. Ha egy ágyást valaki már beültetett, akkor más már nem ültetett oda. A munka eredményét tárolja el a `szinek.txt` fájlban! A fájlban soronként az ágyások sorszámának sorrendjében két értéket, az ágyás színét jelző karaktert és az ültetéshez tartozó felajánlás sorszámát írja ki. Amelyik ágyás virág nélkül maradt, színként a # karaktert, a felajánlás sorszámának helyére pedig a 0-t írja!

Digitális kultúra
emelt szint

Azonosító jel:

Példa a szöveges kimenetek kialakításához:

```
2. feladat
A felajánlások száma: 465

3. feladat
A bejárat mindkét oldalán ültetők: 10 34 98 107 115 142 156 160
340 360 378

4. feladat
Adja meg az ágyás sorszámát! 100
A felajánlók száma: 8
A virágágyás színe, ha csak az első ültet: Z
A virágágyás színei: O Z S K

5. feladat
Átszervezéssel megoldható a beültetés.
```

A szinek.txt fájl 1-3. sora:

```
K 10
K 10
K 10
```

A szinek.txt fájl 268-270. sora:

```
Z 364
# 0
# 0
```

50 pont

Források:

Dobogókő

https://turistautak.openstreetmap.hu/mentettutv-163532779823na Utolsó letöltés: 2021.11.17

Állóképesség

https://www.netfit.eu/public/pb_riport_megye.php Utolsó letöltés: 2021.12.1.
https://www.ksh.hu/stadat?lang=hu&theme=okt Utolsó letöltés: 2021.12.1.

```
"""
2022. október digitkult: Virágágyások
@author Klemand66
"""

print("\n1. feladat\n")

betxt = open("felajanlas.txt")

sor=betxt.readline()
db=int(sor.strip()) # az ágyások száma

felajanlasok=[]
for sor in betxt:
    d=sor.strip().split()
    adatsor=[int(d[0]),int(d[1]),d[2]]
          #  0: első  1: utolsó  2: szín
    felajanlasok.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A felajánlások száma: {len(felajanlasok)}')



print("\n3. feladat\n")

mindketton=list(filter(lambda i: felajanlasok[i][0]>felajanlasok[i][1] ,range(len(felajanlasok))))
sorszamlista=list(map(lambda i: str(i+1), mindketton))
print(f'A bejárat mindkét oldalán ültetők:\n{" ".join(sorszamlista)}')



print("\n4. feladat\n")

agyas=int(input(f'Adja meg az ágyás sorszámát (1 <= sorszám <= {db}): ').strip())
aktfelajanlasok=list(filter(lambda f: (f[0] <= agyas <= f[1]) \
                or (f[1]<f[0] and (agyas>=f[0] or agyas<=f[1])),felajanlasok))
print(f'A felajánlók száma: {len(aktfelajanlasok)}')
if len(aktfelajanlasok)>0:
    print(f'A virágágyás színe, ha csak az első ültet: {aktfelajanlasok[0][2]}')
else:
    print("Ezt az ágyást nem ültetik be.")

if len(aktfelajanlasok)>0:
    szinlista=[]
    for a in aktfelajanlasok:
        if a[2] not in szinlista:
            szinlista.append(a[2])
    print(f'A virágágyás színei: {" ".join(szinlista)}')
```

```
print("\n5. feladat\n")

agyasdbk=[]
for i in range(db):
    agyasdbk.append(0)

for f in felajanlasok:
    if f[0]<=f[1]:
        for i in range(f[0],f[1]+1):
            agyasdbk[i]+=1
    if f[0]>f[1]:
        for i in range(f[0],len(f)):
            agyasdbk[i]+=1
        for i in range(f[1]+1):
            agyasdbk[i]+=1

if 0 not in agyasdbk:
    print("Minden ágyás beültetésére van jelentkező.")
elif sum(agyasdbk)>=db:
    print("Átszervezéssel megoldható a beültetés.")
else:
    print("A beültetés nem oldható meg.")



print("\n6. feladat\n")

kitxt = open("szinek.txt","w")


szinlista, beultetolista = [], []
for i in range(db):
    szinlista.append("#")
    beultetolista.append(0)

for n,f in enumerate(felajanlasok):
    if f[0]<=f[1]:
        for i in range(f[0],f[1]):
            if szinlista[i] == "#":
                szinlista[i]=f[2]
                beultetolista[i]=n+1
    if f[0]>f[1]:
        for i in range(f[0],db):
            if szinlista[i] == "#":
                szinlista[i]=f[2]
                beultetolista[i]=n+1
        for i in range(f[1]):
            if szinlista[i] == "#":
                szinlista[i]=f[2]
                beultetolista[i]=n+1

for i,sz in enumerate(szinlista):
    kitxt.write(f'{sz} {beultetolista[i]}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================ RESTART: C:\Python\KlEmProgPy\EmeltDigKult2022okt\EmeltDigKult2022okt.py ================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A felajánlások száma: 465

3. feladat

A bejárat mindkét oldalán ültetők:
10 34 98 107 115 142 156 160 340 360 378

4. feladat

Adja meg az ágyás sorszámát (1 <= sorszám <= 2222): 100
A felajánlók száma: 8
A virágágyás színe, ha csak az első ültet: Z
A virágágyás színei: Z S O K

5. feladat

Átszervezéssel megoldható a beültetés.

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

A szinek.txt szövegfile:

K 10	Z 232	# 0	B 115
K 10	Z 232	# 0	B 115
K 10	Z 232	# 0	B 115
K 10	Z 232	# 0	B 115
K 10	Z 232	# 0	B 115
K 10	Z 232	# 0	B 115
K 10	...	# 0	B 115
K 10	L 22	# 0	B 115
K 10	L 22	# 0	B 115
K 10	L 22	# 0	B 115
K 10	L 22	# 0	B 115
K 10	N 77	# 0	B 115
K 10	Z 364	# 0	L 34
K 10	Z 364	# 0	L 34
K 10	Z 364	# 0	L 34
K 33	Z 364	L 427	L 34
K 33	Z 364	L 427	L 34
K 33	Z 364	L 427	L 34
L 34	Z 364	L 427	L 34
L 34	Z 364	L 427	L 34
L 34	# 0	L 427	K 10
L 34	# 0	L 427	K 10
L 34	# 0	L 427	K 10
L 34	# 0	L 427	K 10
P 58	# 0	...	K 10
P 58	# 0	B 115	K 10
P 58	# 0	B 115	K 10
P 58	# 0	B 115	K 10
P 58	# 0	B 115	K 10

2023. május: RGB színek

Informatika emelt szint | Azonosító jel: | | | | | | | | | | | | | | |

4. RGB színek

Fényképek számítógépes tárolására sokféle képformátumot ismerünk. A legegyszerűbb típusok az eredeti kép pixeleihez tartozó színértékeket tárolják, például ilyen a BMP vagy a fényképezőgépek által használt RAW formátum. Az ilyen képállomány bináris, melynek első néhány bájtja a kép jellemzőit írja le, míg a további bájtok sorfolytonosan a kép egy-egy pixelének színét adják meg.

Ebben a feladatban egy RAW formátumú fájlból átalakított szöveges állományt kell feldolgoznia. A `kep.txt` szöveges állomány egy 640×360 méretű, RGB kódolású képet ír le. Az állomány csak a képpontok színét tartalmazza sorfolytonosan, azaz a fájl 360 sorának mindegyike 640 képpontból, képpontonként három színértékből áll. Az első szám a piros (**R**ed), a második szám a zöld (**G**reen) és a harmadik szám a kék (**B**lue) értéket tartalmazza. Az RGB értékek 0 és 255 közötti egészek, melyeket a fájlban egy-egy szóköz választ el.

Részlet az állomány első néhány sorából (az alábbi példákban szereplő számhármasok félkövér betűstílussal vannak kiemelve):

```
0 85 112 0 86 113 0 86 113 0 86 113 0 86 113 0 86 113 …
0 86 111 0 86 111 0 86 113 0 86 113 0 86 113 0 86 113 …
0 86 113 0 86 113 0 86 113 0 86 113 0 86 113 1 87 114 …
0 86 113 0 86 113 0 86 113 0 86 113 1 87 114 1 87 114 …
…
```

Az első sor első három száma a kép bal felső képpontjának színe, azaz RGB(0, 85, 112), míg a második sor harmadik pixelének színe RGB(0, 86, 113), a negyedik sor hatodik oszlopában lévő képpont színe RGB(1, 87, 114).
A szöveges állomány által leírt kép:

Készítsen programot, amely az állomány adatait felhasználva megoldja az alábbi feladatokat! A program forráskódját mentse `rgb` néven! A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, és feltételezheti, hogy a beolvasandó adatok a leírtaknak megfelelnek.

A képernyőre írást igénylő részfeladatok esetén – a mintához tartalmában hasonlóan – írja ki a képernyőre a feladat sorszámát (például: `2. 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 a `kep.txt` állomány tartalmát, és tárolja el a 640×360 képpont színét!
2. Kérje be a felhasználótól a kép egy pontjának sor- és oszlopszámát (a számozás mindkét esetben 1-től indul), és írja a képernyőre az adott képpont RGB színösszetevőit a minta szerint!

Informatika emelt szint | Azonosító jel: | | | | | | | | | | | | | | | |

3. Világosnak tekintjük az olyan képpontot, amely RGB-értékeinek összege 600-nál nagyobb. Számolja meg és írja ki, hogy a teljes képen hány világos képpont van!
4. A kép legsötétebb pontjainak azokat a pontokat tekintjük, amelyek RGB-értékeinek összege a legkisebb. Adja meg, hogy mennyi a legkisebb összeg, illetve keresse meg az ilyen RGB összegű pixeleket, és írja ki mindegyik színét RGB(r,g,b) formában a mintának megfelelően!
5. A képen a kék ég látható közepén egy felhővel. Az ég és a felhő színe között jelentős különbség van, így az ég-felhő határvonal programmal is felismerhető. Ennek megtalálásához készítsen függvényt `hatar` néven, amely megadja, hogy egy adott sorban van-e olyan hely a képen, ahol az egymás melletti képpontok kék színösszetevőinek eltérése meghalad egy adott értéket! A függvény kapja meg paraméterként a sor számát, illetve az eltérés értékét, melyek egészek! A függvény visszatérési értéke egy logikai érték legyen, amely megadja, hogy az adott sorban volt-e az eltérést meghaladó különbség az egymás melletti képpontok kék színében!
6. Keresse meg a képen a felhő első és utolsó sorát az előzőleg elkészített függvény segítségével úgy, hogy eltérésként 10-et ad meg a függvénynek bemenetként! Adja meg az első és az utolsó olyan sor sorszámát, ahol az eltérés a soron belül valahol 10-nél nagyobb!

Példa a szöveges kimenetek kialakításához:

```
2. feladat:
Kérem egy képpont adatait!
Sor:180
Oszlop:320
A képpont színe RGB(184,183,181)
3. feladat:
A világos képpontok száma: 7837
4. feladat:
A legsötétebb pont RGB összege: 197
A legsötétebb pixelek színe:
RGB(0,85,112)
RGB(0,86,111)
RGB(0,86,111)
6. feladat:
A felhő legfelső sora: 103
A felhő legalsó sora: 280
```

45 pont

```
"""
2023. május: RGB színek
@author Klemand67
"""


print("\n1. feladat\n")

"""
A könnyebb feldolgozás kedvéért a képpontokat nem 640x360-as táblázatban,
hanem sorfolytonos listában tároljuk.
A transzformációk:
sor=index//640 +1, oszlop=index%640 +1
index=(sor-1)*640+(oszlop-1)
"""

betxt = open("kep.txt")

szinek=[]

for sor in betxt:
    darsor=sor.strip().split()
    adatsor=list(map(int,darsor))
    for i in range(0,640*3,3):
        szin=[adatsor[i],adatsor[i+1],adatsor[i+2]]
        szinek.append(szin)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print("Kérem egy képpont adatait!")
s=int(input("Sor: "))
o=int(input("Oszlop: "))
ind=(s-1)*640+(o-1)
aktszin=szinek[ind]

print(f'A képpont színe RGB({aktszin[0]},{aktszin[1]},{aktszin[2]})')



print("\n3. feladat\n")

# Algoritmussal

vdb=0
for szin in szinek:
    if sum(szin)>600:
        vdb+=1

print(f'Algoritmussal: A világos képpontok száma: {vdb} ')

# Python-függvényekkel

vilagosak=list(filter(lambda szin: sum(szin)>600, szinek))

print(f'\nPython-függvényekkel: A világos képpontok száma: {len(vilagosak)} ')
```

```
print("\n4. feladat\n")

osszegek=list(map(sum,szinek))
ls=min(osszegek)

print(f'A legsötétebb pont RGB összege: {ls}')

legsotetebbek=list(filter(lambda szin: sum(szin)==ls, szinek))

print("A legsötétebb pixelek színe:")
for szin in legsotetebbek:
    print(f'RGB({szin[0]},{szin[1]},{szin[2]})')



print("\n5. feladat\n")

def hatar(s,d):
    for i in range((s-1)*640,(s-1)*640+360-1):
        if abs(szinek[i+1][2]-szinek[i][2])>d:
            return True
    return False

print(f'Az elkészített függvény:\n\n\
def hatar(s,d):\n\
    for i in range((s-1)*640,(s-1)*640+360-1):\n\
        if abs(szinek[i+1][2]-szinek[i][2])>d:\n\
            return True\n\
    return False')



print("\n6. feladat\n")

felho=list(filter(lambda s: hatar(s,10), range(360)))

print(f'A felhő legfelső sora: {felho[0]}')
print(f'A felhő legalsó sora: {felho[-1]}')


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================ RESTART: C:\Python\KlEmProgPyFlap\EmeltInfo2023maj\EmeltInfo2023maj.py ================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Kérem egy képpont adatait!
Sor: 180
Oszlop: 320
A képpont színe RGB(184,183,181)

3. feladat

Algoritmussal: A világos képpontok száma: 7837

Python-függvényekkel: A világos képpontok száma: 7837

4. feladat

A legsötétebb pont RGB összege: 197
A legsötétebb pixelek színe:
RGB(0,85,112)
RGB(0,86,111)
RGB(0,86,111)

5. feladat

Az elkészített függvény:

def hatar(s,d):
    for i in range((s-1)*640,(s-1)*640+360-1):
        if abs(szinek[i+1][2]-szinek[i][2])>d:
            return True
    return False

6. feladat

A felhő legfelső sora: 103
A felhő legalsó sora: 280


A befejezéshez nyomd meg az ENTER billentyűt!
```

2023. május idegennyelvű: Szállítószalag

Informatika emelt szint | Azonosító jel:

4. Szállítószalag

Egy speciális, kör alakú üzemcsarnokban különböző anyagok szállításához körpályán mozgó szállítószalagot állítottak üzembe. Erre a szalagra bárhol feltehetnek bármilyen anyagot egy rekeszbe zárva, annak beállításával, hogy hova szánják. A célhelyen a szalagról automatikusan lekerül a rekesz. A szalag az óramutató járásával egyező irányban mozog, akkor indul el, ha rekesz kerül rá és addig mozog, amíg van rajta szállítandó rekesz. Az ilyen szállítási feladatokat kell feldolgoznia.

Rendelkezésére áll a *szallit.txt* nevű adatfájl, amelynek első sorában két egész szám található, egymástól egy szóközzel elválasztva. Az első a szállítószalag hossza, a második a szállítószalag egy egységnyi elmozdulásához szükséges idő. A fájl további legfeljebb 1000 sorában soronként négy egész szám található, egymástól szóközzel elválasztva. Az első szám megadja, hogy mikor tették a szalagra a rekeszt. A következő kettő megmutatja, hogy honnan hova tart a szállítás (a megtett út a szalag hosszánál mindig kevesebb). Az utolsó pedig a szállítandó tömeget jelenti. A hosszúságot és a pozíciókat távolságegységben, az időt időegységben, a tömeget pedig tömegegységben adták meg. Az adatfájl egyetlen adata sem nagyobb 500-nál.

Például:

```
200 3
1 134 64 34
14 22 129 83
14 135 54 21
23 31 40 61
…
```

A példában a fájl első 5 sora látható. Az első sor mutatja, hogy 200 egység hosszú a szalag, és egy egységnyi távolságot 3 időegység alatt tesz meg. A 2. sor tartalmazza az első rekesz adatait. A 3. sor szerint a 14. időegységben a 22-es hosszúságegységnél felhelyeznek egy rekeszt, amely a 129-es hosszúságegységnél kerül le. A rekesz 83 egység tömegű. A rekesz tehát 129-22=107 egység utat tesz meg, és a 14+107*3=335-ös időegységben ér célba. A következő sorban szereplő rekesz szintén a 14. időegységben kerül a szalagra, a 135-ös hosszúságegységtől az 54-es hosszúságegységig halad. A megtett távolság ezért a 0 hosszúságegységig 200-135 egység, azt követően pedig 54 egység, azaz összesen 119 egységnyi távolságot tesz meg a 21 egység tömegű rekesz.

Készítsen programot, amely az állomány adatait felhasználva az alábbi kérdésekre válaszol! A program forráskódját mentse *szalag* 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: `2. 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 a *szallit.txt* állomány adatait, és annak felhasználásával oldja meg a következő feladatokat!
2. Kérje be egy szállítás sorszámát, majd írassa ki annak indulási és célhelyét! (A szállításokat 1-től sorszámozzuk.)

Informatika
emelt szint

Azonosító jel:

3. Készítsen függvényt `tav` néven, amely megadja a szállítás távolságát a szalag hosszának, valamint az indulási és a célhelynek ismeretében! A függvényt használja fel a későbbi feladatok megoldása során. A függvényfejet az alábbiaknak megfelelően készítse el, megoldásában az ott szereplő változóneveket használja!

```
Függvény tav(szalaghossz, indulashelye, erkezeshelye : egész szám):
egész szám
```

4. Határozza meg, hogy a rendelkezésre álló szállítások során mekkora volt a legnagyobb szállítási távolság! Írja a képernyőre a maximális távolságot és az összes ilyen hosszúságú szállítás sorszámát!

5. Adja meg, mekkora tömeg haladt el összesen a 0 pozíciójú hely előtt! Az onnan induló vagy oda érkező rekeszeket ne vegye figyelembe!

6. Kérjen be egy időpontot, és határozza meg az adott időpontban szállított rekeszek sorszámát! Az éppen akkor induló rekeszeket vegye figyelembe, de a célba érőket ne! Ha nem volt szállított rekesz, akkor a rekeszek sorszáma helyett az „üres” szót írja ki!

7. Hozza létre a *tomeg.txt* fájlt, amely megadja, hogy az egyes helyekről összességében mennyi tömeget szállítottak el! Azok a helyek ne jelenjenek meg a fájlban, ahonnan nem történt szállítás! (A fájlba írt adatok sorrendje tetszőleges.)

Példa a szöveges kimenetek kialakításához:

```
2. feladat
Adja meg, melyik adatsorra kíváncsi! 3
Honnan: 135 Hova: 54

4. feladat
A legnagyobb távolság: 195
A maximális távolságú szállítások sorszáma: 31 33

5. feladat
A kezdőpont előtt elhaladó rekeszek össztömege: 957

6. feladat
Adja meg a kívánt időpontot! 300
A szállított rekeszek halmaza: 1 2 3 6 7 10 11
```

Példa az *tomeg.txt* szöveges állomány tartalmára:

```
5 37
12 35
16 26
17 33
18 41
22 83
…
```

45 pont

```
"""
2023. május idegennyelvű: Szállítószalag
@author Klemand67
"""


print("\n1. feladat\n")

betxt = open("szallit.txt")

sor=betxt.readline()
darsor=sor.strip().split()
hossz=int(darsor[0])
eido=int(darsor[1])

rekeszek=[]
for sor in betxt:
    darsor=sor.strip().split()
    adatsor=list(map(int,darsor))
    # 0: idő 1: honnan 2: hova 3: tömeg
    rekeszek.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

ssz=int(input("Adja meg, melyik adatsorra kíváncsi! "))

print(f'Honnan: {rekeszek[ssz-1][1]} Hova: {rekeszek[ssz-1][2]}')



print("\n3. feladat\n")

def tav(hossz,honnan,hova):
    if hova>honnan:
        return hova-honnan
    else:
        return hossz-honnan+hova

print(f'Az elkészített függvény:\n\n\
def tav(hossz,honnan,hova):\n\
    if hova>honnan:\n\
        return hova-honnan\n\
    else:\n\
        return hossz-honnan+hova')



print("\n4. feladat\n")

tavok=list(map(lambda rekesz: tav(hossz,rekesz[1],rekesz[2]), rekeszek))
maxtav=max(tavok)

print(f'A legnagyobb távolság: {maxtav}')

maxindexek=list(filter(lambda i: tavok[i]==maxtav, range(len(tavok))))
maxsorszamok=list(map(lambda i: str(i+1), maxindexek))

print(f'A maximális távolságú szállítások sorszáma: {" ".join(maxsorszamok)}')
```

```
print("\n5. feladat\n")

elhaladok=list(filter(lambda rekesz: rekesz[1]>rekesz[2]>0, rekeszek))
ehtomegek=list(map(lambda rekesz: rekesz[3], elhaladok))
ossztomeg=sum(ehtomegek)

print(f'A kezdőpont előtt elhaladó rekeszek össztömege: {ossztomeg}')



print("\n6. feladat\n")

aktido=int(input("Adja meg a kívánt időpontot! "))

aktindexek=list(filter(lambda i: \
rekeszek[i][0]<=aktido<rekeszek[i][0]+tavok[i]*eido, range(len(rekeszek))))
# \: hosszú kódsor törése
aktsorszamok=list(map(lambda i: str(i+1), aktindexek))

if len(aktsorszamok)>0:
    print(f'A szállított rekeszek halmaza: {" ".join(aktsorszamok)}')
else:
    print(f'A szállított rekeszek halmaza: üres')



print("\n7. feladat\n")

helyek=[]
tomegek=[]

for rekesz in rekeszek:
    if rekesz[1] in helyek:
        helyindex=helyek.index(rekesz[1])
        tomegek[helyindex]+=rekesz[3]
    else:
        helyek.append(rekesz[1])
        tomegek.append(rekesz[3])


kitxt = open("tomeg.txt","w")

for i,hely in enumerate(helyek):
    kitxt.write(f'{hely} {tomegek[i]}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================ RESTART: C:\Python\KlEmProgPyFlap\EmeltInfo2023mid\EmeltInfo2023mid.py ================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Adja meg, melyik adatsorra kíváncsi! 3
Honnan: 135 Hova: 54

3. feladat

Az elkészített függvény:

def tav(hossz,honnan,hova):
    if hova>honnan:
        return hova-honnan
    else:
        return hossz-honnan+hova

4. feladat

A legnagyobb távolság: 195
A maximális távolságú szállítások sorszáma: 31 33

5. feladat

A kezdőpont előtt elhaladó rekeszek össztömege: 957

6. feladat

Adja meg a kívánt időpontot! 300
A szállított rekeszek halmaza: 1 2 3 6 7 10 11

7. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

A befejezéshez nyomd meg az ENTER billentyűt!
```

Az tomeg.txt szövegfile:

```
134 149
22 83
135 90
31 61
161 78
26 44
69 48
12 35
18 41
199 38
140 47
152 57
81 90
129 90
112 40
98 38
17 33
16 26
75 88
110 62
42 22
5 37
38 69
95 33
153 29
46 67
90 31
96 61
92 77
```

2023. május, digitális kultúra: Ütemezés

Digitális kultúra
emelt szint

Azonosító jel:

3. Ütemezés

A diákok hasznos nyári időtöltését biztosítják a különböző nyári táborok. Egy iskolai osztályban felmérték, kik melyiken vennének részt szívesen. Ebben a feladatban az előzetes igényfelmérés adatait dolgozzuk fel.

A `taborok.txt` tabulátorokkal tagolt fájl soronként egy tábor adatait tartalmazza. Az állomány legfeljebb 100 sorból áll. Az első két számpár a tábor első és utolsó napjának dátuma. A számpárok első értéke a hónap, a második a nap sorszáma. Mivel a nyári szünet programjairól van szó, csak június 16. és augusztus 31. közötti dátumok szerepelhetnek. Ezt követik azon diákok betűjelei, akik érdeklődnek a tábor iránt, végül a tábor témája olvasható. A diákok betűjele az angol ábécé egy nagybetűs karaktere, a tábor témája egyetlen ékezetmentes szó.

Részlet az állomány első néhány sorából:

```
6   26  7   10  GIOSY  foci
7   14  7   21  FPUY   szinjatszo
7   27  8   2   DKPRX  hittan
7   28  8   6   FJLOP  cserkesz
7   9   7   14  FKO    gombasz
…
```

A második sorban egy színjátszó tábor adatai szerepelnek. A tábor július 14-én kezdődik és július 21-én fejeződik be, tehát összesen 8 napos, és ez az F, P, U, Y betűjelekkel azonosított tanulókat érdekli.

Készítsen programot, amely az állomány adatait felhasználva megoldja az alábbi feladatokat! A program forráskódját mentse `utemez` néven! A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, és feltételezheti, hogy a beolvasandó adatok a leírtaknak megfelelnek.

A képernyőre írást igénylő részfeladatok esetén – a mintához tartalmában hasonlóan – írja ki a képernyőre a feladat sorszámát (például: `2. 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 `taborok.txt` állomány tartalmát!
2. Jelenítse meg a képernyőn, hogy hány tábor adatait tartalmazza a bemeneti fájl! Írja a képernyőre az elsőként és az utolsóként rögzített tábor témáját!
3. Írja a képernyőre, mikor kezdődik a „***zenei***” tábor! Ha több ilyen tábor is volt, az összeset jelenítse meg a lenti mintának megfelelően! Ha egy sem volt, akkor a „***Nem volt zenei tábor.***” szöveget jelenítse meg a képernyőn!
4. Keresse meg, melyik táborba jelentkeztek a legtöbben! Írja a képernyőre a tábor kezdő dátumát és a témáját! Ha több ilyen tábor is van, az összeset jelenítse meg!
5. Készítsen függvényt `sorszam` néven, amely megadja, hogy a paraméterként kapott hónap és nap a nyári szünet hányadik napja! A dátumot a függvény két egész számként kapja meg, a visszaadott érték egy egész szám legyen! A nyári szünet első napja június (6. hó) 16. A nyári szünet 77. napja augusztus (8. hó) 31. (A nyári hónapok rendre 30, 31, 31 naposak.) A későbbi feladatok megoldásánál ezt a függvényt felhasználhatja.
6. Kérjen be a felhasználótól egy dátumot a lenti mintának megfelelően, majd adja meg, hány tábor zajlik éppen ekkor!

Digitális kultúra
emelt szint

Azonosító jel:

7. Olvassa be egy tanuló betűjelét! Határozza meg, hogy az adott betűjelű tanuló mely táborok iránt érdeklődött! A táborok adatait kezdő dátum szerint növekvő sorrendben írja az *egytanulo.txt* fájlba a minta formátumának megfelelően! Jelenítse meg a képernyőn, hogy a tanuló mindegyiken részt tud-e venni, azaz nincs-e olyan nap, amelyen több táborban kellene lennie!

Az *egytanulo.txt* fájl első néhány sora:

```
6.18-6.29. evezos
6.22-6.26. cserkesz
7.2-7.8. csillagasz
7.8-7.20. erdojaro
...
```

Példa a szöveges kommunikáció kialakításához:

```
2. feladat
Az adatsorok száma: 28
Az először rögzített tábor témája: foci
Az utoljára rögzített tábor témája: filmes
3. feladat
Zenei tábor kezdődik 8. hó 4. napján.
Zenei tábor kezdődik 6. hó 18. napján.
4. feladat
Legnépszerűbbek:
8 27 fotos
6. feladat
hó: 8
nap: 1
Ekkor éppen 3 tábor tart.
7. feladat
Adja meg egy tanuló betűjelét: L
Nem mehet el mindegyik táborba.
```

50 pont

Források:

A feladatlap bázisszövegei az eredeti forrásszövegek módosításával (rövidítésével, nyelvtani egyszerűsítésével), adatainak felhasználásával, de az eredeti szöveg, adatok integritásának megtartása mellett jöttek létre. Az eredeti szövegek, adatok, képek forrása:

1A. Dominó
https://szamoldki.hu/hu/kalkulator/dominos-konny-feladvany Utolsó letöltés: 2021.11.18.

1B. Adóazonosító jel
https://www.jogiforum.hu/torvenytar/tv/1996/XX/ Utolsó letöltés: 2022.06.17.

2. Szinkron
Internetes Szinkron Adatbázis, https://iszdb.hu/. Utolsó letöltés: 2022.09.10.

```
"""
2023. május digitkult: Ütemezés
@author Klemand67
"""

print("\n1. feladat\n")

betxt = open("taborok.txt")

taborok=[]
for sor in betxt:
    d=sor.strip().split("\t")
    adatsor=[int(d[0]),int(d[1]),int(d[2]),int(d[3]),list(d[4]),d[5] ]
         #  0: ehó  1: enap  2: uhó 3: unap 4: érdeklődők listája 5: téma
    taborok.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'Az adatsorok száma: {len(taborok)}')
print(f'Az először rögzített tábor témája: {taborok[0][5]}')
print(f'Az utoljára rögzített tábor témája: {taborok[-1][5]}')



print("\n3. feladat\n")

zenei=list(filter(lambda tabor: tabor[5]=="zenei", taborok))

if len(zenei)>0:
    for z in zenei:
        print(f'Zenei tábor kezdődik {z[0]}. hó {z[1]}. napján.')
else:
    print("Nem volt zenei tábor.")



print("\n4. feladat\n")

print("Legnépszerűbbek:")

nepszerusegek=list(map(lambda tabor: len(tabor[4]), taborok))
maxnsz=max(nepszerusegek)
legnepszerubbek=list(filter(lambda tabor: len(tabor[4])==maxnsz, taborok))

for tabor in legnepszerubbek:
    print(f'{tabor[0]} {tabor[1]} {tabor[5]}')
```

```
print("\n5. feladat\n")

def sorszam(ho,nap):
    if ho==6:
        return nap-15
    elif ho==7:
        return nap+15
    else:
        return nap+46

print(f'Az elkészített függvény:\n\n\
def sorszam(ho,nap):\n\
    if ho==6:\n\
        return nap-15\n\
    elif ho==7:\n\
        return nap+15\n\
    else:\n\
        return nap+46')



print("\n6. feladat\n")

aktho=int(input("hó: "))
aktnap=int(input("nap: "))

tartanak=list(filter(lambda t: \
sorszam(t[0],t[1])<=sorszam(aktho,aktnap)<=sorszam(t[2],t[3]), taborok))

print(f'\nEkkor éppen {len(tartanak)} tábor tart:')

# szorgalmi, felsoroljuk, melyek ezek
for tabor in tartanak:
    print(tabor[5], end=" ")
print()



print("\n7. feladat\n")

akttan=input("Adja meg egy tanuló betűjelét: ")

akttaborok=list(filter(lambda tabor: akttan in tabor[4], taborok))
akttaborok.sort(key=lambda tabor: sorszam(tabor[0],tabor[1]))

kitxt = open("egytanulo.txt","w")

for t in akttaborok:
    kitxt.write(f'{t[0]}.{t[1]}-{t[2]}.{t[3]}. {t[5]}\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

# Az ütközésekhez felhasználjuk a kezdő dátum szerinti rendezést
utkozesek=list(filter(lambda i: \
sorszam(akttaborok[i-1][2],akttaborok[i-1][3])>=sorszam(akttaborok[i][0],akttaborok[i][1]), \
range(1,len(akttaborok))))

#szorgalmi: megadjuk az ütközések számát is
if len(utkozesek)>0:
    print(f'Nem mehet el mindegyik táborba, {len(utkozesek)} ütközés van.')
else:
    print("Elmehet mindegyik táborba.")


input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
============== RESTART: C:\Python\KlEmProgPyFlap\EmeltDigKult2023maj\EmeltDigKult2023maj.py ==============

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

Az adatsorok száma: 28
Az először rögzített tábor témája: foci
Az utoljára rögzített tábor témája: filmes

3. feladat

Zenei tábor kezdődik 8. hó 4. napján.
Zenei tábor kezdődik 6. hó 18. napján.

4. feladat

Legnépszerűbbek:
8 27 fotos

5. feladat

Az elkészített függvény:

def sorszam(ho,nap):
    if ho==6:
        return nap-15
    elif ho==7:
        return nap+15
    else:
        return nap+46

6. feladat

hó: 8
nap: 1

Ekkor éppen 3 tábor tart:
hittan cserkesz korus

7. feladat

Adja meg egy tanuló betűjelét: L

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

Nem mehet el mindegyik táborba, 11 ütközés van.

A befejezéshez nyomd meg az ENTER billentyűt!
```

Az egytanulo.txt szövegfile:

```
6.18-6.29. evezos
6.22-6.26. cserkesz
7.2-7.8. csillagasz
7.8-7.20. erdojaro
7.8-7.14. cukrasz
7.11-7.22. fuvos
7.15-7.28. kornyezettudatos
7.28-8.6. cserkesz
8.9-8.14. hegymaszo
8.13-8.18. torna
8.15-8.18. uszas
8.16-8.24. filmes
8.24-8.31. regesz
8.27-8.30. fotos
```

2023. október: Társas

Informatika emelt szint | Azonosító jel: | | | | | | | | | | | | | | |

4. Társas

Mi sem izgalmasabb a gyerekeknek, mint a nagyszülők szekrényeiben való kutakodás. Egy ilyen alkalommal az öt unoka egy régi dobozra bukkant. A doboz egy társasjáték kellékeit rejtette: a játék tábláját, egy dobókockát és hat különböző színű, lúd formájú bábut. A táblán több ösvényen lehet eljuttatni a ludat az itatóhoz. A játékosok a kezdés előtt kiválasztják a közös ösvényt, amelyen haladni fognak.

A ludak annyi mezőt ugorhatnak előre, amennyit a játékosok dobnak a kockával.

Az ösvényen lehetnek különleges mezők, amelyekre érkezve:

- a lúd előreugorhat még egyszer annyit, mint amennyit dobott a játékos,
- vissza kell térnie arra a mezőre, amelyről indította az ugrást.

Az `osvenyek.txt` fájl soronként egy ösvény adatait tartalmazza. Az ösvények sorszámozása 1-től kezdődik. Az állomány legfeljebb 50 sorból, soronként legfeljebb 250 karakterből áll. Minden sorban egy-egy karaktersorozat van. A karakterek az M, E, V nagybetűk. Az M karakter egy mindenféle különleges tulajdonságtól mentes mezőt jelent. Ha valaki az E karakterű mezőre lép, előreugorhat még ugyanannyit, mint amennyivel ide jutott. Ha a V karakterű mezőre lép, a bábunak vissza kell térnie oda, ahonnan ide ugrott. Két egymást követő különleges karakter között mindig legalább 6 darab M karakter van.

A játék akkor ér véget, ha mindenki ugyanannyiszor dobott, és legalább egy játékos elérte az utolsó mezőt, vagy túllépett rajta. Azok a nyertesek, akik elérték az utolsó mezőt vagy túlléptek rajta.

A `dobasok.txt` fájl egyetlen, szóközökkel tagolt sorból áll, amely a kockával egymást követően dobott értékeket tartalmazza. A fájlban 1 és 6 közötti számokból legfeljebb 2000 található. Tudjuk, hogy ennyi dobás biztosan elegendő a játék befejezéséhez. Azt nem rögzítettük, hogy melyik értéket melyik játékos dobta.

Részlet a `dobasok.txt` állomány első sorából:

```
2 4 6 5 6 1 …
```

Ha két játékos játszik, akkor dobásaik értékei felváltva szerepelnek a `dobasok.txt` állományban. Tehát az első játékos először 2-t dob, majd a második játékos 4-et, utána az első játékos 6-ot, a második játékos 5-öt, és így tovább.

Részlet az `osvenyek.txt` állomány első néhány sorából:

```
MMMMMMMMMM…M
MEMMMMMMEMMMMMMMMMM…
…
```

Ha a második ösvényen játszanak (az MEMM kezdetűn), akkor az első játékos 2-es dobásával a második mezőre lép. Ez E típusú mező, ezért azonnal, dobás nélkül még két mezővel előrelép, így jut az első dobásával először a kettes, majd a négyes mezőre. Ezt követően a második játékos a 4-es dobásával a negyedik mezőre lép. Ez M típusú, azaz mindenféle különleges tulajdonságtól mentes, így játékosunk a következő dobásig itt marad. Ismét az első játékos jön. Ő a 6-os dobásával a tizedik mezőre léphet tovább. Ez M típusú, ezért nem lép innen el, a második játékos jön. Ő az 5-ös dobásával a negyedik mezőről a kilencedik mezőre lép. Ez E típusú, tehát rögtön előreléphet a dobásának megfelelően újabb öt mezőt. A következő ábra ezeket a lépéseket szemlélteti:

mező sorszáma	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
játékos1		1		1						2						3	
mező típusa	M	E	M	M	M	M	M	M	E	M	M	M	M	M	M	M	M
játékos2				1					2					2	3		

Készítsen programot, amely az állományok adatait felhasználva megoldja az alábbi feladatokat! A program forráskódját mentse `lud` néven! A program megírásakor a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, és feltételezheti, hogy a beolvasandó adatok a leírtaknak megfelelnek.

Informatika emelt szint | Azonosító jel:

A képernyőre írást igénylő részfeladatok esetén – a mintához tartalmában hasonlóan – írja ki a képernyőre a feladat sorszámát (például: `5. feladat`), és utaljon a kiírt tartalomra is! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Mindkét esetben elfogadott az ékezetmentes kiírás is.

1. Olvassa be és tárolja el az *osvenyek.txt* és a *dobasok.txt* állományok tartalmát!
2. Jelenítse meg a képernyőn, hogy hány ösvény adatait tartalmazza az *osvenyek.txt* fájl, és mennyi dobás szerepel a *dobasok.txt* fájlban!
3. Határozza meg, hogy melyik ösvény áll a legtöbb mezőből, és jelenítse meg az ösvény sorszámát és a mezők számát! Ha több ilyen is van, elegendő egyet megjelenítenie.
4. Olvassa be és tárolja el egy ösvény sorszámát és a játékot játszók számát (legalább 2, legfeljebb 5)! A későbbiekben ezekkel az adatokkal dolgozzon!
5. Készítsen statisztikát a megadott sorszámú ösvény mezőiből! Jelenítse meg, hogy ez milyen típusú mezőből mennyit tartalmaz! Ha egy adott típusú mező nem szerepel, akkor azt ne jelenítse meg! (Megoldása teszteléséhez használja az első három ösvényt is, ezek ugyanis nem tartalmaznak minden karaktert!)
6. Írja a *kulonleges.txt* fájlba, hogy a választott ösvény mely mezői különlegesek! Soronként egy mezőt adjon meg a mező sorszámával és a mező típusát megadó karakterrel! A két értéket egy tabulátor karakterrel válassza el egymástól!
7. Határozza meg, hogy melyik játékos jutna a legmesszebb, ha a választott ösvényen minden mező M típusú lenne! Jelenítse meg a legtávolabb jutó(k) sorszámát és azt, hogy a dobások hányadik körében alakult ki a végeredmény! Ha több ilyen játékos van, elegendő csak egyet megjelenítenie.
8. Határozza meg, ki nyer, ha figyelembe veszi a mezők típusát is! Jelenítse meg a nyertes(ek) sorszámát és azt, hogy a többi bábu milyen sorszámú mezőn áll az utolsó teljes kör végén!

45 pont

Példa a szöveges kommunikáció kialakításához:

```
2. feladat
A dobások száma: 1956
Az ösvények száma: 43

3. feladat
Az egyik leghosszabb a(z) 11. ösvény, hossza: 206

4. feladat
Adja meg egy ösvény sorszámát! 9
Adja meg a játékosok számát! 5

5. feladat
M: 185 darab
V: 8 darab
E: 8 darab

7. feladat
A játék a(z) 54. körben fejeződött be. A legtávolabb jutó(k) sorszáma: 5

8. feladat
Nyertes(ek): 4 5
A többiek pozíciója:
1. játékos, 153. mező
2. játékos, 185. mező
3. játékos, 183. mező
```

```
"""
2023. október: Társas
@author Klemand67
"""

print("\n1. feladat\n")

betxt = open("osvenyek.txt")

osvenyek=[]

for sor in betxt:
    osvenyek.append(list(sor.strip()))

betxt.close()

betxt = open("dobasok.txt")

sor=betxt.readline()
darsor=sor.strip().split()
dobasok=list(map(int,darsor))

betxt.close()

print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A dobások száma:  {len(dobasok)}')
print(f'Az ösvények száma:  {len(osvenyek)}')



print("\n3. feladat\n")

osvhosszak=list(map(lambda osv: len(osv), osvenyek))
maxhossz=max(osvhosszak)

maxindexek=list(filter(lambda i: osvhosszak[i]==maxhossz, range(len(osvhosszak))))
maxsorszamok=list(map(lambda i: str(i+1), maxindexek))

print(f'A leghosszabb a(z) {maxsorszamok[0]}. ösvény, hossza: {maxhossz}')

print("\nSzorgalmi: az összes maximális hosszúságú ösvény megadása")
print(f'A leghosszabb(ak) a(z) {" ".join(maxsorszamok)} ösvény(ek), hossza(uk): {maxhossz}\n')



print("\n4. feladat\n")

ssz=int(input("Adja meg egy ösvény sorszámát! "))
n=int(input("Adja meg a játékosok számát (legalább 2, legfeljebb 5)! "))
```

```
print("\n5. feladat\n")

aktosv=osvenyek[ssz-1]
tipus=["M","V","E"]
tipusdb=[]

for i in range(3):
    szures=list(filter(lambda t: t==tipus[i], aktosv))
    tipusdb.append(len(szures))

for i in range(3):
    if tipusdb[i]>0:
        print(f'{tipus[i]}: {tipusdb[i]} darab')



print("\n6. feladat\n")

kitxt = open("kulonleges.txt","w")

for i,o in enumerate(aktosv):
    if o!="M":
        kitxt.write(f'{i+1}\t{o}\n')

kitxt.close()
print("A kiíratás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n7. feladat\n")

jatekosokdobasai=[]

for i in range(n):
    indexszures=list(filter(lambda j: j%5==i, range(len(dobasok))))
    dobasszures=list(map(lambda j: dobasok[j], indexszures))
    jatekosokdobasai.append(dobasszures)

jatekosokakthelyei=[]
maxhely=len(aktosv)

for i in range(n):
    aktdobasok=jatekosokdobasai[i]
    akthelyek=[]
    honnan=0
    d=0
    while True:
        hova=honnan+aktdobasok[d]
        akthelyek.append(hova)
        honnan=hova
        d+=1
        if hova>maxhely:
            break
    jatekosokakthelyei.append(akthelyek)

lepesek=[]

for i in range(n):
    lepesek.append(len(jatekosokakthelyei[i]))

minlepes=min(lepesek)
```

```
nyertesek=list(filter(lambda i: lepesek[i]==minlepes, range(len(lepesek))))
nyertessorszamok=list(map(lambda i: str(i+1), nyertesek))

print(f'A játék a(z) {d}. körben fejeződött be. A legtávolabb jutó(k) sorszáma: {nyertessorszamok[0]}')

print("\nSzorgalmi: az összes legtávolabb jutó játékos megadása")
print(f'A játék a(z) {d}. körben fejeződött be. A legtávolabb jutó(k) sorszáma: {" ".join(nyertessorszamok)}\n')



print("\n8. feladat\n")

jatekosokakthelyei=[]

for i in range(n):
    aktdobasok=jatekosokdobasai[i]
    akthelyek=[]
    honnan=0
    d=0
    while True:
        hova=honnan+aktdobasok[d]
        if hova<maxhely:
            if aktosv[hova-1]=="E":
                hova+=aktdobasok[d]
            if aktosv[hova-1]=="V":
                hova=honnan
        akthelyek.append(hova)
        honnan=hova
        d+=1
        if hova>maxhely:
            break
    jatekosokakthelyei.append(akthelyek)

lepesek=[]

for i in range(n):
    lepesek.append(len(jatekosokakthelyei[i]))

minlepes=min(lepesek)

nyertesek=list(filter(lambda i: lepesek[i]==minlepes, range(len(lepesek))))
nyertessorszamok=list(map(lambda i: str(i+1), nyertesek))

print(f'Nyertesek: {" ".join(nyertessorszamok)}\n')
print("A többiek pozíciója:")

for i in range(n):
    if lepesek[i]>minlepes:
        print(f'{i+1}. játékos, {jatekosokakthelyei[i][minlepes-1]}. mező')

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
================ RESTART: C:/Python/KlEmProgPyFlap/EmeltInfo2023okt/EmeltInfo2023okt.py ================

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A dobások száma:  1956
Az ösvények száma:  43

3. feladat

A leghosszabb a(z) 11. ösvény, hossza: 206

Szorgalmi: az összes maximális hosszúságú ösvény megadása
A leghosszabb(ak) a(z) 11 ösvény(ek), hossza(uk): 206


4. feladat

Adja meg egy ösvény sorszámát! 9
Adja meg a játékosok számát (legalább 2, legfeljebb 5)! 5

5. feladat

M: 185 darab
V: 8 darab
E: 8 darab

6. feladat

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

7. feladat

A játék a(z) 54. körben fejeződött be. A legtávolabb jutó(k) sorszáma: 5

Szorgalmi: az összes legtávolabb jutó játékos megadása
A játék a(z) 54. körben fejeződött be. A legtávolabb jutó(k) sorszáma: 5


8. feladat

Nyertesek: 4 5

A többiek pozíciója:
1. játékos, 153. mező
2. játékos, 185. mező
3. játékos, 183. mező

A befejezéshez nyomd meg az ENTER billentyűt!
```

A kulonleges.txt szövegfile:

```
7       V
16      V
25      E
45      E
66      E
83      V
90      V
97      E
108     V
128     V
140     V
149     E
161     V
168     E
179     E
191     E
```

2023. október, digitális kultúra: Reklám

Digitális kultúra
emelt szint

Azonosító jel:

3. Reklám

Egy termék hirdetésének hatékonyságát vizsgálták három, egymáshoz hasonló lélekszámú városban. Egy 30 napos időszak középső tíz napján zajlott reklámkampány a három város közül kettőben. Az egyik városban a helyi televízióban reklámozták a terméket, a másik városban utcai plakáton hirdették, mindkét városban az időszak 11-edik napjától a 20-adik napjáig. A harmadik városban nem volt reklámkampány, illetve az előbbi két városban sem volt az időszak első 10 és az utolsó 10 napján.

A kampány sikerességét a 30 nap során a három városból leadott rendelések alapján mérték. Ezek az adatok megtalálhatók a *rendel.txt* szöveges állományban, melynek néhány sora a következő:

```
13 PL 2
13 PL 1
13 TV 1
13 NR 1
13 TV 3
```

A rendelések időrendben vannak, minden sorban egy-egy rendelés szerepel. Egy soron belül az első szám a rendelés napja, a következő két betű azt a várost jelöli, ahol a rendelést leadták, míg a harmadik elem a termékből rendelt darabszám. Az adatokat a soron belül szóközök választják el egymástól. A fenti példában látható rendelések mind a 13. napon történtek. Az első két rendelést abban a városban adták le, ahol plakáton hirdettek (PL). A harmadik és ötödik rendelést abban a városban adták le, ahol a helyi televízióban hirdettek (TV). A negyedik rendelést abban a városban adták le, ahol nem volt reklámkampány (NR). Az első példában látható rendeléssel 2 terméket, az utolsó rendeléssel 3-at, míg a középső három rendelés mindegyikével 1–1 terméket rendeltek.

Egy-egy napon általában több rendelés érkezett, és tudjuk, hogy minden nap rendeltek valamelyik városból, de nem feltétlenül rendeltek minden nap mindhárom városból. Az összes rendelések száma 1000-nél kevesebb, és minden rendelésben a darabszám 10-nél kisebb.

Készítsen programot, amely a három városból származó rendelések adatait felhasználva az alábbi feladatokat oldja meg! A program forráskódját mentse *reklam* néven! A program megírásakor a bemeneti állományban található, vagy a felhasználó által megadott adatok helyességét, érvényességét nem kell ellenőriznie, feltételezheti, hogy azok a leírtaknak megfelelnek.

A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: `2. feladat`) és utaljon a megjelenítendő tartalomra! Ha a felhasználótól kér be adatot, jelenítse meg a képernyőn, hogy milyen értéket vár! Az ékezetmentes kiírás is elfogadott a képernyőn történő megjelenítés és a szöveges állományba való írás során.

1. Olvassa be és tárolja el a további feldolgozáshoz a *rendel.txt* állomány tartalmát!
2. Állapítsa meg, hogy hány rendelés történt a teljes időszakban, és írja a képernyőre a rendelések számát!
3. Kérje be a felhasználótól egy nap számát, és adja meg, hogy hány rendelés történt az adott napon!
4. Számolja meg, hogy hány nap nem volt rendelés a reklámban nem érintett városból, és írja ki a napok számát! Ha egy ilyen nap sem volt, akkor írja ki „`Minden nap volt rendelés a reklámban nem érintett városból`” szöveget!

Digitális kultúra
emelt szint

Azonosító jel:

5. Állapítsa meg, hogy mennyi volt az egy rendelésben szereplő legnagyobb darabszám, és melyik volt az a nap, amikor az első ilyen számú rendelést leadták! Az eredményt a lenti minta szerint írja ki!

6. Készítsen függvényt `osszes` néven, amely megadja, hogy mennyi volt egy adott városból egy adott napon a rendelt termékek száma! A függvény bemenete a három város egyikére utaló kétbetűs szöveg és a nap sorszáma legyen. Amennyiben szükséges, akkor további paramétert is felvehet a rendelések adatainak elérése érdekében. A függvény visszaadott értéke a rendelt darabszámok összege legyen! A függvényt például a következő módon lehessen meghívni: `osszes("PL", 7)`. A függvényt a későbbiekben felhasználhatja a további feladatok megoldásakor.

7. Számítsa ki, hogy a kampány utáni első napon, azaz a 21-edik napon melyik városból mennyit rendeltek a termékből! Az eredményt a lenti mintának megfelelő formában írja ki!

8. Összesítse városonként, hogy hány rendelés érkezett az első 10, a 11-20-adik valamint a záró 10 napon! Az eredményt (a fejlécet is beleértve) táblázatos formában, tabulátorokkal tagoltan jelenítse meg a képernyőn, illetve írja azonos formátumban a *kampany.txt* szöveges állományba!

Minta a felhasználóval történő kommunikációhoz:

```
2. feladat:
A rendelések száma: 971
3. feladat:
Kérem, adjon meg egy napot: 9
A rendelések száma az adott napon: 27
4. feladat:
3 nap nem volt a reklámban nem érintett városból rendelés
5. feladat:
A legnagyobb darabszám: 9, a rendelés napja: 22
7. feladat:
A rendelt termékek darabszáma a 21. napon PL: 43 TV: 36 NR: 18
8. feladat:
Napok    1..10  11..20  21..30
PL       98     159     106
TV       97     143     100
NR       91     86      91
```

A feladat a következő oldalon folytatódik.

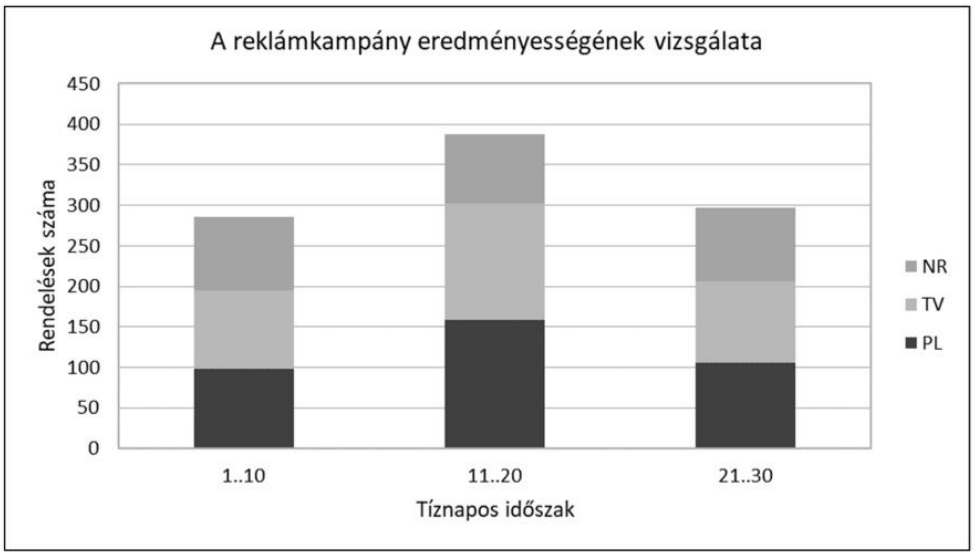
Digitális kultúra
emelt szint

Azonosító jel:

9. Készítsen táblázatkezelő alkalmazás segítségével halmozott oszlopdiagramot az előző feladatban kiszámított adatok felhasználásával! Töltse be a `kampany.txt` szöveges állományból az adatokat, vagy ha nem tudta megoldani az előző feladatot, akkor gépelje be a mintában szereplő értékeket! A városok sorrendje a mintától eltérő is lehet!

10. A diagram címét, jelmagyarázatát és tengelyfeliratait a minta szerint alakítsa ki! A munkafüzetet mentse `kampany` néven a táblázatkezelő program alapértelmezett formátumában!

Minta a diagramról:



50 pont

```
"""
2023. okt digitkult: Reklám
@author Klemand67
"""

print("\n1. feladat\n")

betxt = open("rendel.txt")

rendelesek=[]

for sor in betxt:
    d=sor.strip().split()
    adatsor=[int(d[0]),d[1],int(d[2])]
        #  0: nap  1: város  2: darab
    rendelesek.append(adatsor)

betxt.close()
print("A beolvasás és a szövegfájl lezárása sikeresen befejeződött.")



print("\n2. feladat\n")

print(f'A rendelések száma: {len(rendelesek)}')



print("\n3. feladat\n")

aktnap=int(input("kérem, adjon meg egy napot (1 és 30 között): "))

aktrendelesek=list(filter(lambda rendeles: rendeles[0]==aktnap, rendelesek))

print(f'A rendelések száma az adott napon: {len(aktrendelesek)}')



print("\n4. feladat\n")

NRnapok=[]

for n in range(30):
    NRnapok.append(0)

for r in rendelesek:
    if r[1]=="NR":
        NRnapok[r[0]-1]+=1

kimaradonapok=list(filter(lambda i: NRnapok[i]==0, range(len(NRnapok))))

if len(kimaradonapok)==0:
    print("Minden nap volt rendelés a reklámban nem érintett városból.\n")
else:
    print(f'{len(kimaradonapok)} nap nem volt a reklámban nem érintett városból rendelés.\n')
    # Szorgalmi
    kimaradonapokstr=list(map(lambda i: str(i+1), kimaradonapok))
    print(f'Szorgalmi feladat: Ezek a napok: {" ".join(kimaradonapokstr)}\n')
```

```
print("\n5. feladat\n")

maxdb=rendelesek[0][2]
maxnap=rendelesek[0][0]

for i in range(1,len(rendelesek)):
    if rendelesek[i][2]>maxdb:
        maxdb=rendelesek[i][2]
        maxnap=rendelesek[i][0]

print(f'A legnagyobb darabszám: {maxdb}, a(z első) rendelés napja {maxnap}')



print("\n6. feladat\n")

def osszes(varos,nap):
    aktrendelesek=list(filter(lambda rendeles: rendeles[0]==nap and rendeles[1]==varos, rendelesek))
    dbk=list(map(lambda rendeles: rendeles[2], aktrendelesek))
    return sum(dbk)

print(f'Az elkészített függvény:\n\n\
def osszes(varos,nap):\n\
    aktrendelesek=list(filter(lambda rendeles: rendeles[0]==nap and rendeles[1]==varos, rendelesek))\n\
    dbk=list(map(lambda rendeles: rendeles[2], aktrendelesek))\n\
    return sum(dbk)')



print("\n7. feladat\n")

print(f'A rendelt termékek darabszáma a 21. napon \
PL: {osszes("PL",21)} TV: {osszes("TV",21)}  NR: {osszes("NR",21)} ')



print("\n8. feladat\n")

print(f'A rendelések számának összesítése városonként és időszakonként\n')

def darab(varos,nap):
    aktrendelesek=list(filter(lambda rendeles: rendeles[0]==nap and rendeles[1]==varos, rendelesek))
    return len(aktrendelesek)

varosok=["PL","TV","NR"]
osszrendelesek=[]

for varos in varosok:
    varosrendelesek=[]
    for szakasz in range(3):
        reszrendelesek=0
        for nap in range(1,11):
            reszrendelesek+=darab(varos,szakasz*10+nap)
        varosrendelesek.append(reszrendelesek)
    osszrendelesek.append(varosrendelesek)
```

```
kitxt = open("kampany.txt","w")

print(f'Napok\t1..10\t11..20\t21..30')
kitxt.write(f'Napok\t1..10\t11..20\t21..30\n')

for i, varos in enumerate(varosok):
    print(f'{varos}\t{osszrendelesek[i][0]}\t{osszrendelesek[i][1]}\t{osszrendelesek[i][2]}')
    kitxt.write(f'{varos}\t{osszrendelesek[i][0]}\t{osszrendelesek[i][1]}\t{osszrendelesek[i][2]}\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

print(f'Szorgalmi feladat: A rendelt termékek számának összesítése városonként és időszakonként\n')

varosok=["PL","TV","NR"]
mindosszesek=[]

for varos in varosok:
    varososszesek=[]
    for szakasz in range(3):
        reszosszesek=0
        for nap in range(1,11):
            reszosszesek+=osszes(varos,szakasz*10+nap)
        varososszesek.append(reszosszesek)
    mindosszesek.append(varososszesek)

kitxt = open("kampanyszorgalmi.txt","w")

print(f'Napok\t1..10\t11..20\t21..30')
kitxt.write(f'Napok\t1..10\t11..20\t21..30\n')

for i, varos in enumerate(varosok):
    print(f'{varos}\t{mindosszesek[i][0]}\t{mindosszesek[i][1]}\t{mindosszesek[i][2]}')
    kitxt.write(f'{varos}\t{mindosszesek[i][0]}\t{mindosszesek[i][1]}\t{mindosszesek[i][2]}\n')

kitxt.close()
print("\nA kiíratás és a szövegfájl lezárása sikeresen befejeződött.\n")

input("\nA befejezéshez nyomd meg az ENTER billentyűt!")
```

```
============== RESTART: C:/Python/KlEmProgPyFlap/EmeltDigKult2023okt/EmeltDigKult2023okt.py ==============

1. feladat

A beolvasás és a szövegfájl lezárása sikeresen befejeződött.

2. feladat

A rendelések száma: 971

3. feladat

kérem, adjon meg egy napot (1 és 30 között): 9
A rendelések száma az adott napon: 27

4. feladat

3 nap nem volt a reklámban nem érintett városból rendelés.

Szorgalmi feladat: Ezek a napok: 10 11 30


5. feladat

A legnagyobb darabszám: 9, a(z első) rendelés napja 22

6. feladat

Az elkészített függvény:

def osszes(varos,nap):
    aktrendelesek=list(filter(lambda rendeles: rendeles[0]==nap and rendeles[1]==varos, rendelesek))
    dbk=list(map(lambda rendeles: rendeles[2], aktrendelesek))
    return sum(dbk)

7. feladat

A rendelt termékek darabszáma a 21. napon PL: 43 TV: 36  NR: 18

8. feladat

A rendelések számának összesítése városonként és időszakonként

Napok   1..10   11..20  21..30
PL      98      159     106
TV      97      143     100
NR      91      86      91

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.

Szorgalmi feladat: A rendelt termékek számának összesítése városonként és időszakonként

Napok   1..10   11..20  21..30
PL      278     439     304
TV      287     383     278
NR      254     229     255

A kiíratás és a szövegfájl lezárása sikeresen befejeződött.


A befejezéshez nyomd meg az ENTER billentyűt!
```

A kampany.txt szövegfile:

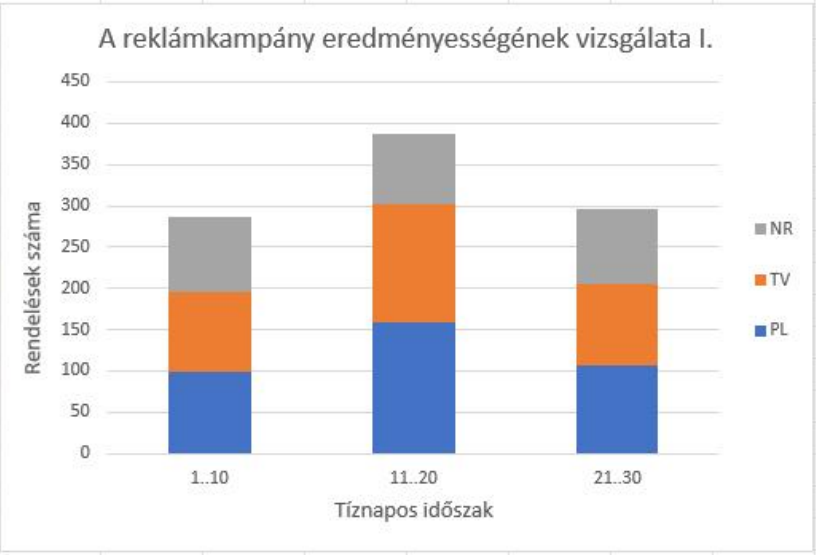
```
Napok   1..10   11..20  21..30
PL      98      159     106
TV      97      143     100
NR      91      86      91
```

A kampanyszorgalmi.txt szövegfile:

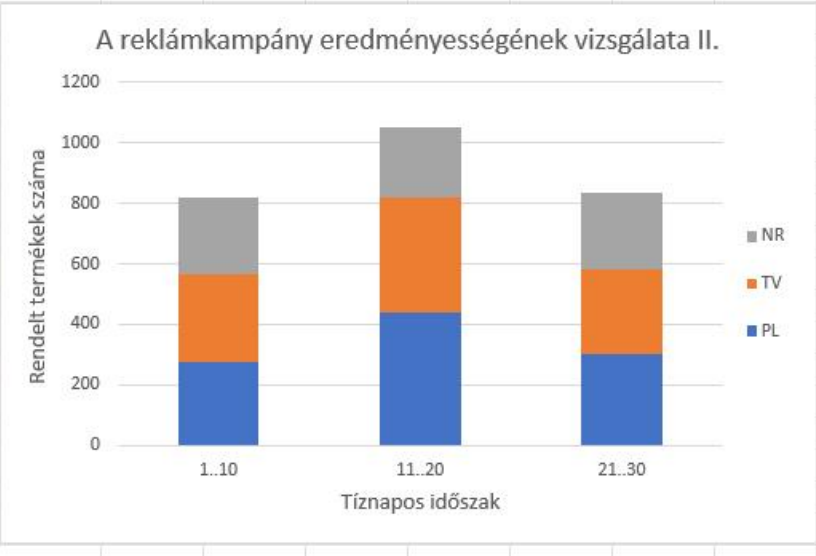
```
Napok   1..10   11..20  21..30
PL      278     439     304
TV      287     383     278
NR      254     229     255
```

Az ekészített diagramok:

	A	B	C	D
1	Napok	1..10	11..20	21..30
2	PL	98	159	106
3	TV	97	143	100
4	NR	91	86	91

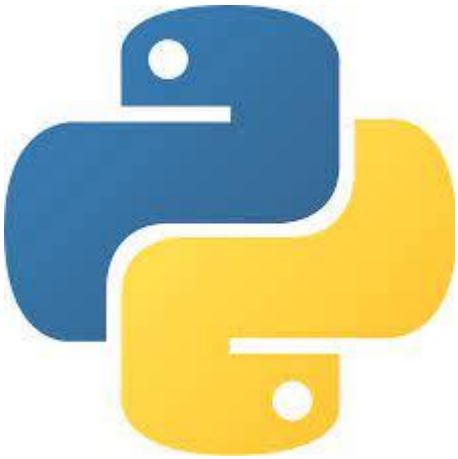


	A	B	C	D
1	Napok	1..10	11..20	21..30
2	PL	278	439	304
3	TV	287	383	278
4	NR	254	229	255



Előzmény: Bevezető gyakorlatok az emelt szintű informatika érettségi programozási feladataihoz Python nyelven

Bevezető gyakorlatok az emelt szintű informatika érettségi programozási feladataihoz Python nyelven



Klement András

2022-23