```
void gyors( int tomb[], int bal, int jobb )
{
  if ( bal < jobb )
  {
    int also = bal, felso = jobb + 1, kulcs = tomb[ bal ];
    for ( ; ; )
    {
      while ( ++also < felso && tomb[ also ] < kulcs )
        ;
      while ( tomb[ --felso ] > kulcs )
        ;
      if ( also >= felso )
        break;
      csere( tomb, also, felso );
    }
    csere( tomb, felso, bal );
    gyors( tomb, bal, felso - 1 );
    gyors( tomb, felso + 1, jobb );
  }
}
```

```
int binaris( int tomb[], int meret, int ertek )
{
  int also = 0, felso = meret - 1;
  while ( also <= felso )
  {
    int kozepso = ( also + felso ) / 2;
    if ( tomb[ kozepso ] == ertek )
      return kozepso;
    if ( tomb[ kozepso ] > ertek )
      felso = kozepso - 1;
    else
      also = kozepso + 1;
  }
  return -1;
}
```

```
void shell2( int tomb[], int meret )
{
  int lk[] = { 6, 3, 1 };
  int lkindex;
  for ( lkindex = 0; lkindex < sizeof( lk ) / sizeof( int ); ++lkindex )
  {
    int lepeskoz = lk[ lkindex ];
    int j;
    for ( j = lepeskoz; j < meret; ++j )
    {
      int i = j - lepeskoz;
      int kulcs = tomb[ j ];
      while ( i >= 0 && tomb[ i ] > kulcs )
      {
        tomb[ i + lepeskoz ] = tomb[ i ];
        i -= lepeskoz;
      }
      tomb[ i + lepeskoz ] = kulcs;
    }
  }
}
```

```
void beszurasos( int tomb[], int meret )
{
  int j;
  for ( j = 1; j < meret; ++j )
  {
    int kulcs = tomb[ j ], i = j - 1;
    while ( i >= 0 && tomb[ i ] > kulcs )
    {
      tomb[ i + 1 ] = tomb[ i ];
      --i;
    }
    tomb[ i + 1 ] = kulcs;
  }
}
```

```
void csere( int tomb[], int i, int j )
{
  int seged = tomb[ i ];
  tomb[ i ] = tomb[ j ];
  tomb[ j ] = seged;
}

void maxkival( int tomb[], int meret )
{
  int j;
  for ( j = meret - 1; j > 0; --j )
  {
    int max = j, i;
    for ( i = 0; i < j; ++i )
      if ( tomb[ i ] > tomb[ max ] )
        max = i;
    csere( tomb, max, j );
  }
}
```

```
int feloszt( int tomb[], int bal, int jobb )
{
  int kulcs = tomb[ jobb ], hatar = bal - 1, akt;
  for ( akt = bal; akt < jobb; ++akt )
    if ( tomb[ akt ] <= kulcs )
      csere( tomb, ++hatar, akt );
  csere( tomb, hatar + 1, jobb );
  return hatar + 1;
}

void gyors2( int tomb[], int bal, int jobb )
{
  if ( bal < jobb )
  {
    int kozepso = feloszt( tomb, bal, jobb );
    gyors2( tomb, bal, kozepso - 1 );
    gyors2( tomb, kozepso + 1, jobb );
  }
}
```

```
void buborek( int tomb[], int meret )
{
  int i, j;
  for ( i = meret - 1; i > 0; --i )
    for ( j = 0; j < i; ++j )
      if ( tomb[ j + 1 ] < tomb[ j ] )
        csere( tomb, j, j + 1 );
}
```

```
void minkival( int tomb[], int meret )
{
  int j;
  for ( j = 0; j < meret - 1; ++j )
  {
    int min = j, i;
    for ( i = j + 1; i < meret; ++i )
      if ( tomb[ i ] < tomb[ min ] )
        min = i;
    csere( tomb, min, j );
  }
}
```