1. feladat: Az alábbi algoritmus egy nem negatív számokat tartalmazó sorozatban olyan elemet ad meg, amely előtti elemek szorzata megegyezik a mögötte levő elemek szorzatával. Írja át időben hatékonyabbra!

```
Eljárás Keresés(VAN,K):
  VAN:=Hamis
  Ciklus i=1-től N-ig
    Ha szorzat(1,i-1)=szorzat(i+1,N) akkor VAN:=Igaz; K:=i
  Ciklus vége
Eljárás vége.

Függvény Szorzat(A,B):egész
  S:=1
  Ciklus j=A-tól B-ig
    S:=S*X[j]
  Ciklus vége
  Szorzat:=S
Függvény vége.
```

2. feladat: Adott N diák M feladatban elért pontszáma egy N*M-es mátrixban. (Egy feladatra kapott pontszám -3 és +3 közötti érték.) Írja át az alábbi algoritmust gyorsabbra, amely kiválogatja az alábbi módon meghatározott minimális pontszámot elért diákok sorszámát!

```
Eljárás Lassu:
  i:=1
  Ciklus amíg i≤N és osszpont(i)=0
    i:=i+1
  Ciklus vége
  Ha i>N akkor
    db:=0 [azaz Mindenki kiesett]
  különben
    minpont:=osszpont(i)
    Ciklus j=i-től N-ig
      Ha osszpont(j)>0 és osszpont(j)<minpont akkor minpont:=osszpont(j)
    Ciklus vége
    db:=0
    Ciklus i=1-től N-ig
      Ha osszpont(i)=minpont akkor db:=db+1; minek[db]:=i
    Ciklus vége
  Elágazás vége
Eljárás vége.

Függvény Osszpont(i:egész):egész
  ossz:=0
  Ciklus j=1-től M-ig
    Ha pontok[i,j]<0 akkor  pont:=0
    különben ha pontok[i,j]>0 akkor  pont:=pontok[i,j]
    különben ha pontok[i,j]=0 akkor  pont:=0
    ossz:=ossz+pont
  Ciklus vége
  Osszpont:=ossz
Függvény vége.
```

Megoldás

1. feladat:

A legfőbb probléma az, hogy folyamatosan újraszámolja a szorzatot ahelyett, hogy felhasználná a már kiszámolt rész-szorzatot; ill. kihasználatlanul hagyja az elágazásbeli két szorzat „egymást kiegészítő” voltát. Ki lehet használni, hogy a két szorzatra a ciklus i-edik lépésében igaz, hogy

*balSzorzat**X[i]**jobbSzorzat* = szorzat(1,N).

A másik gond a keresés tétel „szabálytalan” alkalmazása.

Az első probléma megoldása után az algoritmus:

```
Eljárás Keresés(VAN,K):
  szB:=1; szJ:=szorzat(2,N) [előfeldolgozás]
  VAN:=Hamis
  Ciklus i=1-től N-ig
    Ha szB=szJ akkor VAN:=Igaz; K:=i
    szB:=szB*X[i]; szJ:=szJ Div X[i]
  Ciklus vége
Eljárás vége.
```

A második probléma megoldása után:

```
Eljárás Keresés(VAN,K):
  szB:=1; szJ:=szorzat(2,N) [előfeldolgozás]
  VAN:=Hamis
  i:=1
  Ciklus i≤N és nem VAN
    VAN:=szB=szJ
    szB:=szB*X[i]; szJ:=szJ Div X[i]
    i:+1
  Ciklus vége
  Ha VAN akkor K:=i-1
Eljárás vége.
```

… és két apróbb optimalizálással az `i`-korrekciótól és a `VAN` ciklusbeli újraszámolásától is megszabadulunk:

```
Eljárás Keresés(VAN,K):
  szB:=1; szJ:=szorzat(2,N) [előfeldolgozás]
  i:=1
  Ciklus i≤N és szB≠szJ
    szB:=szB*X[i]; szJ:=szJ Div X[i]
    i:+1
  Ciklus vége
  VAN:=i≤N
  Ha VAN akkor K:=i
Eljárás vége.
```

A kódolást most csak a lényegi részre végezzük el:

```
Procedure Kereses(Var VAN:Boolean; Var K:Integer);
  Var
    szB,szJ,i:Integer;
Begin
  szB:=1; szJ:=szorzat(2,N) {előfeldolgozás}
  i:=1;
  While (i<=N) and (szB<>szJ) do
  Begin
    szB:=szB*X[i]; szJ:=szJ Div X[i];
    Inc(i);
  End;
  VAN:=i<=N;
  If VAN then K:=i;
End;
```

```
Function Szorzat(Const A,B:Integer):Integer;
  Var
    j:Integer;
Begin
  S:=1;
  For j:=A to B do
  Begin
    S:=S*X[i];
  End;
  Szorzat:=S
End;
```

Megjegyzések:

1. Az előbbi megoldás ugyan hatékonyabb, de szigorúan véve **nem** jó megoldás, mivel nem pont ugyanazt adja eredményül, mint a kiinduló. Ugyanis az eredeti a több megoldás közül az utolsót választja. Alakítsa át úgy a fenti megoldást, hogy ő is az **utolsót** határozza meg!
2. Ennél még jobb megoldás is található! Ötlete az, hogy egyszer számoljuk ki az összes `i`-hez tartozó megelőző és követő szorzatokat, értelemszerűen egy-egy tömbben tárolva, majd a lényegi eljárásban a `Szorzat` függvény helyett ezekre a tömbökre hivatkozunk. Annyival jobb, hogy osztást nem tartalmaz.

2. feladat:

Az algoritmus problémás jellemzői:

- az `Osszpont` függvénybeli 3-ágú elágazás valójába egy `Ha-akkor` utasítás (üres `különben` ággal); a segéd `pont` változó megspórolható;
- a `Lassú` eljárásban az `Összpont` függvény hívása többször történik (mint elegendő lenne);
- a minimum-kiválasztás tétel „szabálytalan” alkalmazása;
- a megszerzett információk nem felhasználása.

```
Eljárás Lassu:
  i:=1
  Ciklus amíg i≤N és osszpont(i)=0
    i:=i+1
  Ciklus vége
  Ha i>N akkor
    db:=0 [azaz Mindenki kiesett]
  különben
    minpont:=osszpont(i)
    Ciklus j=i+1-től N-ig
      osszJ:=osszpont(j)
      Ha osszJ>0 és osszJ<minpont akkor minpont:=osszJ
    Ciklus vége
    db:=0
    [tudjuk, hogy i-ig 0 az összpontszám!]
    Ciklus j=i+1-től N-ig
      Ha osszpont(j)=minpont akkor db:+1; minek[db]:=j
    Ciklus vége
  Elágazás vége
Eljárás vége.
```

… további észrevétel:

- az összpont számítása még így is jó néhány (`i`-n túli) indexre duplán történik.

```
Eljárás Lassu:
  i:=1
  Ciklus amíg i≤N és osszpont(i)=0
    i:=i+1
  Ciklus vége
  Ha i>N akkor
    db:=0 [azaz Mindenki kiesett]
  különben
    minpont:=osszpont(i)
```

```
    Ciklus j=i+1-től N-ig
      osszJ:=osszpont(j)
      ossz[j]:=osszJ [előfeldolgozás: megjegyezzük az ossz[1..N] tömbben]
      Ha osszJ>0 és osszJ<minpont akkor minpont:=osszJ
    Ciklus vége
    db:=0
    [tudjuk, hogy i-ig 0 az összpontszám; és ezért
     az ossz tömbben a hozzájuk tartozó értéket nem állítottuk be!]
    Ciklus j=i+1-től N-ig
      Ha ossz[j]=minpont akkor db:+1; minek[db]:=j
    Ciklus vége
  Elágazás vége
Eljárás vége.
```

… és a hatékonyabb összpontszámítás:

```
Függvény Osszpont(i:egész):egész
  ossz:=0
  Ciklus j=1-től M-ig
    Ha pontok[i,j]>0 akkor ossz:+pontok[i,j]
  Ciklus vége
  Osszpont:=ossz
Függvény vége.
```

A kódolással most egyáltalán nem törődünk. Egyetlen dologra azonban felhívjuk a figyelmet: a programban kétféle célra szerepel az `ossz` azonosító. Az `Osszpont` függvényben egy skalár, a `Lassú` (most már talán nevezhetnénk `Gyors`nak is) eljárásban egy a `pontok` mátrix sorainak számával azonos hosszúságú tömb.

Megjegyzés: egy másik megoldásban azzal kerüljük el az `Osszpont` függvény többszörös hívását, hogy összevonjuk a minimum-kiválasztást és a kiválogatást.

```
Eljárás Gyors:
   [az első nem 0 összpontú megkeresése:]
   i:=1
   Ciklus amíg i≤N és osszpont(i)=0
     i:=i+1
   Ciklus vége
   Ha i>N akkor [mindenki 0 pontos]
     db:=0
   különben [van nem 0 pontos, az 1. éppen ilyen]
     [a nem 0 minimum pont kiválasztása, és egyben kiválogatása:]
     minpont:=osszpont(i)
     db:=1; minek[db]:=i
     Ciklus j=i+1-től N-ig
       pont:=osszpont(j)
       Elágazás
         0<pont és pont<minpont esetén
             minpont:=pont
             db:=1; minek[db]:=i
         pont=minpont esetén
             db:=db+1; minek[db]:=j
       Elágazás vége
     Ciklus vége
   Elágazás vége
Eljárás vége.
```