BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA-INFORMATIKA KAR

Felvételi verseny - minta

Informatika írásbeli

A versenyzők figyelmébe:

1. Minden tömböt 1-től kezdődően indexelünk.
2. A rácstesztekre (A rész) egy vagy több helyes válasz lehetséges. A válaszokat a vizsgadolgozatba írjátok (nem a feladatlapra). Ahhoz, hogy a feltüntetett pontszámot megkapjátok, elengedhetetlenül szükséges, hogy minden helyes választ megadjatok, és kizárólag csak ezeket.
3. A B részben szereplő feladatok megoldásait részletesen kidolgozva a vizsgadolgozatba írjátok.
   - **a.** A feladatok megoldásait *pszeudokódban* vagy egy *programozási nyelvben* (*Pascal/C/C++*) kell megadnotok.
   - **b.** A megoldások értékelésekor az első szempont az algoritmus ***helyessége***, majd a ***hatékonysága***, ami a *végrehajtási időt* és a *felhasznált memória méretét* illeti.
   - **c.** A tulajdonképpeni megoldások előtt, ***kötelezően leírjátok szavakkal az alprogramokat (algoritmusokat), és megindokoljátok a megoldásotok lépéseit***. Feltétlenül írjatok ***megjegyzéseket*** (kommenteket), amelyek segítik az adott megoldás technikai részleteinek megértését. Adjátok meg az azonosítok jelentését és a fölhasznált adatszerkezeteket stb. Ha ez hiányzik, a tételre kapható pontszámotok 10%-kal csökken.
   - **d.** Ne használjatok különleges fejállományokat, előredefiniált függvényeket (például *STL*, karakterláncokat feldolgozó sajátos függvények stb.).

A RÉSZ (60 pont)

A.1. Vajon mit csinál? (6 pont)

A `generál(n)` algoritmus egy ***n*** természetes számot dolgoz fel ($0 < n < 100$).

```
Algoritmus generál(n):
    szám ← 0
    Minden i ← 1, 1801 végezd el
        használt_i ← hamis
    vége(minden)
    Amíg nem használt_n végezd el
        összeg ← 0; használt_n ← igaz
        Amíg n ≠ 0 végezd el
            számjegy ← n MOD 10
            összeg ← összeg + számjegy * számjegy * számjegy
            n ← n DIV 10
        vége(amíg)
        n ← összeg; szám ← szám + 1
    vége(amíg)
    térítsd szám
Vége(algoritmus)
```

Állapítsátok meg a fenti algoritmus hatását.

- **A.** ismételten kiszámítja az ***n*** szám számjegyei köbének összegét, amíg az összeg egyenlővé nem válik az ***n*** számmal és visszatéríti a végrehajtott ismétlések számát
- **B.** kiszámítja az ***n*** szám számjegyei köbének összegét és visszatéríti ezt az összeget
- **C.** kiszámítja az ***n*** szám számjegyei köbének összegét, felülírja ***n*** értékét ezzel az összeggel, és visszatéríti ezt az összeget
- **D.** meghatározza, hogy hányszor lesz felülírva az ***n*** szám a számjegyei köbének összegével, ameddig egy már kiszámolt értéket vagy magát a számot kapja és visszatéríti ezt a számot

A.2. Mely értékekre van szükség? (6 pont)

Adott a `feldolgoz(v, k)` algoritmus, ahol ***v*** egy ***k*** elemű, természetes számokat tároló sorozat ($1 \le k \le 1\,000$).

```
Algoritmus feldolgoz(v, k)
    i ← 1; n ← 0
    Amíg i ≤ k és v_i ≠ 0 végezd el
        y ← v_i, c ← 0
        Amíg y > 0 végezd el
            Ha y MOD 10 > c akkor
                c ← y MOD 10
            vége(ha)
            y ← y DIV 10
        vége(amíg)
        n ← n * 10 + c; i ← i + 1
    vége(amíg)
    térítsd n
Vége(algoritmus)
```

Állapítsátok meg, ***v*** és ***k*** mely értékeire térít vissza az algoritmus 928-at.

A. ***v*** = (194, 121, 782, 0) és ***k*** = 4

B. ***v*** = (928) és ***k*** = 1

C. ***v*** = (9, 2, 8, 0) és ***k*** = 4

D. ***v*** = (8, 2, 9) és ***k*** = 3

A.3. Logikai kifejezés kiértékelése (6 pont)

Adott a ***k*** elemű ***s*** sorozat, amelynek elemei logikai (boolean) típusúak és a `kiértékelés(s, k, i)` algoritmus, ahol ***k*** és ***i*** természetes számok ($0 \leq i \leq k \leq 100$).

```
Algoritmus kiértékelés(s, k, i)
   Ha i ≤ k akkor
      Ha s_i akkor
         térítsd s_i
      különben
         térítsd (s_i vagy kiértékelés(s, k, i + 1))
      vége(ha)
   különben
      térítsd hamis
   vége(ha)
Vége(algoritmus)
```

Határozzátok meg, hányszor hívja meg önmagát a `kiértékelés(s, k, i)` algoritmus a következő programrészletben található hívás következtében:

```
s ← (hamis, hamis, hamis, hamis, hamis, hamis, igaz, hamis, hamis, hamis)
k ← 10, i ← 3
kiértékelés(s, k, i)
```

A. 3-szor

B. ugyanannyiszor, mint a következő programrészlet esetében

```
s ← (hamis, hamis, hamis, hamis, hamis, hamis, hamis, igaz)
k ← 8, i ← 4
kiértékelés(s, k, i)
```

C. 6-szor

D. egyszer sem

A.4. Egyesítés (6 pont)

Adottnak tekintjük az `eleme(x, a, n)` algoritmust, amely eldönti, hogy az ***x*** természetes szám eleme-e az ***n*** elemű ***a*** halmaznak; ***a*** egy ***n*** elemű sorozat, amely egy természetes számokat tartalmazó halmazt ábrázol ($1 \leq n \leq 200$, $1 \leq x \leq 1000$).

Legyenek az alább megadott `egyesítés(x, a, n, b, m, c, p)` és `számol(a, n, b, m, c, p)` algoritmusok, ahol ***a***, ***b*** és ***c*** sorozatok, amelyek természetes számokat tároló és rendre ***n***, ***m*** és ***p*** elemű halmazokat ábrázolnak ($1 \leq n \leq 200$, $1 \leq m \leq 200$, $1 \leq p \leq 400$). A bemeneti paraméterek ***a***, ***n***, ***b*** és ***m***, kimeneti paraméterek pedig ***c*** és ***p***.

```
1. Algoritmus egyesítés(a, n, b, m, c, p):
2.    Ha n = 0 akkor
3.       Minden i ← 1, m végezd el
4.          p ← p + 1, c_p ← b_i
5.       vége(minden)
6.    különben
7.       Ha nem eleme(a_n, b, m) akkor
8.          p ← p + 1, c_p ← a_n
9.       vége(ha)
10.      egyesítés(a, n - 1, b, m, c, p)
11.   vége(ha)
12. Vége(algoritmus)
```

```
1. Algoritmus számol(a, n, b, m, c, p):
2.    p ← 0
3.    egyesítés(a, n, b, m, c, p)
4. Vége(algoritmus)
```

A következő állítások közül melyek bizonyulnak mindig igaznak?

A. ha az ***a*** halmaz egyetlen elemet tartalmaz, az `egyesítés(a, n, b, m, c, p)` algoritmus meghívása végtelen ciklust okoz

B. ha az ***a*** halmaznak négy eleme van, a `számol(a, n, b, m, c, p)` algoritmus meghívása maga után vonja az `egyesítés(a, n, b, m, c, p)` algoritmus 10. sorában található utasítás végrehajtását négyszer

C. ha az ***a*** halmaznak öt eleme van, a `számol(a, n, b, m, c, p)` algoritmus meghívása maga után vonja az `egyesítés(a, n, b, m, c, p)` algoritmus második sorában található utasítás végrehajtását ötször

D. ha az ***a*** és ***b*** halmazok elemei azonosak, az `számol(a, n, b, m, c, p)` algoritmus végrehajtása után a ***c*** halmaznak ugyanannyi eleme lesz, mint az ***a*** halmaznak

A.5. Hatványozás (6 pont)

Melyik alábbi algoritmus számítja ki helyesen a^b értékét, ahol ***a*** és ***b*** természetes számok ($1 \leq a \leq 11$, $0 \leq b \leq 11$)?

A.
```
Algoritmus hatvány(a, b):
   eredmény ← 1
   Amíg b > 0 végezd el
      Ha b MOD 2 = 1 akkor
         eredmény ← eredmény * a
      vége(ha)
      b ←  b DIV 2
      a ← a * a
   vége(amíg)
   térítsd eredmény
Vége(algoritmus)
```

B.
```
Algoritmus hatvány(a, b):
   Ha b ≠ 0 akkor
      Ha b MOD 2 = 1 akkor
         térítsd hatvány(a * a, b / 2) * a
      különben
         térítsd hatvány(a * a, b / 2)
      vége(ha)
   különben
      térítsd 1
   vége(ha)
Vége(algoritmus)
```

C.
```
Algoritmus hatvány(a, b):
   eredmény ← 0
   Amíg b > 0 végezd el
      eredmény ← eredmény * a
      b  ← b - 1
   vége(amíg)
   térítsd eredmény
Vége(algoritmus)
```

D.
```
Algoritmus hatvány(a, b):
   Ha b = 0 akkor
      térítsd 1
   vége(ha)
   térítsd a * hatvány(a, b - 1)
Vége(algoritmus)
```

A.6. Legnagyobb többszörös (6 pont)

Melyik alábbi algoritmus téríti az ***a*** természetes számnak azt a legnagyobb többszörösét, amely kisebb vagy egyenlő a ***b*** természetes számmal ($0 < a < 10\,000$, $0 < b < 10\,000$, $a < b$)?

A.
```
Algoritmus f(a, b):
  c ← b
  Amíg c MOD a = 0 végezd el
    c ← c – 1
  vége(amíg)
  térítsd c
Vége(algoritmus)
```

B.
```
Algoritmus f(a, b):
  Ha a < b akkor
    térítsd f(2 * a, b)
  különben
    Ha a = b akkor
      térítsd a
    különben
      térítsd b
    vége(ha)
  vége(ha)
Vége(algoritmus)
```

C.
```
Algoritmus f(a, b):
  térítsd (b DIV a) * a
Vége(algoritmus)
```

D.
```
Algoritmus f(a, b):
  Ha b MOD a = 0 akkor
    térítsd b
  vége(ha)
  térítsd f(a, b - 1)
Vége(algoritmus)
```

A.7. Adattípus (6 pont)

A következő intervallumok közül melyikhez tartozhat egy ***x*** biten ábrázolt egész adattípussal rendelkező érték (***x*** – szigorúan pozitív természetes szám)?

A. $[0, 2^x]$

B. $[0, 2^{x-1} – 1]$

C. $[-2^{x-1}, 2^{x-1} – 1]$

D. $[-2^x, 2^x – 1]$

A.8. Hívások száma (6 pont)

Legyen az `f(a, b)` algoritmus:

```
Algoritmus f(a, b):
    Ha a > 1 akkor
       térítsd b * f(a - 1, b)
    különben
       térítsd b * f(a + 1, b)
    vége(ha)
Vége(algoritmus)
```

Határozzátok meg, hányszor hívja meg önmagát az `f(a, b)` algoritmus a következő programrészletben található hívás következtében:

```
a ← 4
b ← 3
```

A. 4-szer

B. 3-szor

`c ← f(a, b)`	**C.** végtelenszer **D.** egyszer sem

A.9. Sorozatok (6 pont)

Legyen minden $\boldsymbol{l} \in \{1, 2, 3\}$ hosszú sorozat, amelyeknek elemei az $\{a, b, c, d, e\}$ halmazhoz tartoznak. Hány olyan sorozat található ezek között, amelyeknek elemei növekvően rendezettek és páratlan darab magánhangzót tárolnak (*a* és *e* magánhangzók)?

A. 14
B. 7
C. 81
D. 78

A.10. Pozitív számok (6 pont)

Legyen a `pozitívSzámok(m, a, n, b)` algoritmus:

```
void pozitivSzamok(int m, int a[], int &n, int b[]){
  n = 0;
  for(int i = 1; i <= n; i++){
     if(a[i] > 0){
        n = n + 1;
        b[n] = a[i];
     }
  }
}
```

```
procedure pozitivSzamok(m : integer; a : sir; var
n : integer; var b : sir)
begin
   n := 0;
   for i := 1 to n do
      if a[i] > 0 then begin
         n := n + 1;
         b[n] := a[i];
      end;
end;
```

Mi lesz a hatása a `pozitívSzámok(m, a, n, b)` algoritmus hívásának, ha $\boldsymbol{k} = 4$, $\boldsymbol{x} = (-1, 2, -3, 4)$, $\boldsymbol{p} = -1$ és $\boldsymbol{y} = ()$ (üres sorozat).

A. $\boldsymbol{p} = 3$ és $\boldsymbol{y} = (2, 4)$;
B. $\boldsymbol{p} = 0$ és $\boldsymbol{y} = (2, 4)$;
C. $\boldsymbol{p} = 0$ és $\boldsymbol{y} = ()$;
D. Függ $\boldsymbol{k}$ értékétől

<u>B RÉSZ</u> (30 pont)

B.1. Bűvös számok (15 pont)

Legyen két természetes szám $\boldsymbol{p}$ és $\boldsymbol{q}$ ($2 \le \boldsymbol{p} \le 10$, $2 \le \boldsymbol{q} \le 10$). Egy természetes számot *bűvös*nek nevezünk, ha a $\boldsymbol{p}$ számrendszerben felírt alakjában szereplő számjegyek halmaza azonos a $\boldsymbol{q}$ számrendszerben felírt alakjában szereplő számjegyek halmazával. Például. ha $\boldsymbol{p} = 9$ és $\boldsymbol{q} = 7$, $(31)_{10}$ *bűvös szám*, mivel $(34)_9 = (43)_7$; ha $\boldsymbol{p} = 3$ és $\boldsymbol{q} = 9$, $(9)_{10}$ *bűvös szám*, mivel $(100)_3 = (10)_9$. Adott még a `számjegyek(x, b, c)` alprogram, amely meghatározza az $\boldsymbol{x}$ szám számjegyeit a $\boldsymbol{b}$ számrendszerben (a $\boldsymbol{c}$ sorozatban):

```
Algoritmus számjegyek(x, b, c):
   Amíg x > 0 végezd el
      c[x MOD b] ← 1
      x ← x DIV b
   vége(amíg)
Vége(algoritmus)
```

Követelmények:

a. Írjátok le egy *rekurzív* változatát (ismétlő struktúrák nélkül) a `számjegyek(x, b, c)` algoritmusnak. A fejléce és a hatása legyen azonos a fenti algoritmus fejlécével és hatásával. (**5 pont**)
b. Írjátok le a `számjegyek(x, b, c)` algoritmus rekurzív változatának matematikai modelljét (amelyet kidolgoztatok az a. pontnál). (**3 pont**)
c. Írjatok alprogramot, amely – felhasználva a `számjegyek(x, b, c)` alprogramot – adott $\boldsymbol{p}$ és $\boldsymbol{q}$ számrendszerek ismeretében, meghatározza azt a *bűvös* számokból álló $\boldsymbol{a}$ sorozatot, amely minden 0-nál szigorúan nagyobb és adott $\boldsymbol{n}$ ($1 < \boldsymbol{n} \le 10\,000$) természetes számnál szigorúan kisebb számot tárol. Az alprogram bemeneti paraméterei $\boldsymbol{p}$ és $\boldsymbol{q}$ (a két alap) és az $\boldsymbol{n}$ szám. Kimeneti paraméter az $\boldsymbol{a}$ sorozat és ennek $\boldsymbol{k}$ hossza. (**7 pont**)
Példa: ha $\boldsymbol{p} = 9$, $\boldsymbol{q} = 7$ és $\boldsymbol{n} = 500$, az $\boldsymbol{a}$ sorozatnak $\boldsymbol{k} = 11$ eleme lesz: (1, 2, 3, 4, 5, 6, 31, 99, 198, 248, 297).

B.2. Csokik (15 pont)

Egy reklámcég egy új csokoládét szeretne népszerűsíteni és ebből a célból azt tervezi, hogy ad egy-egy csokit n ($10 \leq n \leq 10^7$) gyermeknek, akiket előbb körbeállítottak. A cég alkalmazottai rájöttek, hogy túl nagy költség lenne, ha minden gyermeknek adnak majd egy csokit. Következésképpen, úgy döntenek, hogy az n gyermek közül csak minden k-adik ($0 < k < n$) fog csokit kapni. Elkezdődik a kiszámolósdi k-ig, majd újból k-ig (amikor az utolsó gyermekhez érnek, a kiszámolás folytatódik az első gyermekkel és így tovább). Számoláskor minden gyermeket figyelembe vesznek, függetlenül attól, hogy kapott már csokit vagy sem. A kiszámolás *leáll, amikor a soron levő csokit egy olyan gyermeknek kellene adni, aki már kapott.*

Követelmények:

a. Adjátok meg azoknak a sorszámoknak a tulajdonságát, amelyek olyan gyermekek sorszámai, akik kapnak csokit. Indokoljátok meg a választ. (**3 pont**)

b. Magyarázzátok el (fogalmazás formájában és matematikai képletekkel) hány gyermek kap csokit. Indokoljátok meg a választ. (**2 pont**)

c. Írjatok alprogramot, amely kiszámítja az ***sz*** számban, hogy hány gyermek *nem* kap csokit. Bemeneti paraméterek az n és k természetes számok, kimeneti paraméter az ***sz*** szám. (**10 pont**)

1. *Példa:* ha $n = 12$ és $k = 9$, akkor ***sz*** = 8 (nem kap csokit az első, a második, a negyedik, az ötödik, a hetedik, a nyolcadik, a tízedik és a 11-edik gyermek).
2. *Példa:* ha $n = 15$ és $k = 7$, akkor ***sz*** = 0 (minden gyermek kap csokit).

Megjegyzések:

1. Minden tétel kidolgozása kötelező.
2. A piszkozatokat nem vesszük figyelembe.
3. Hivatalból jár 10 pont.
4. Rendelkezésetekre áll 3 óra.

JAVÍTÁSI KULCS & MEGOLDÁSOK

HIVATALBÓL ..... **10 pont**

A RÉSZ ..... **60 pont**

A. 1. Vajon mit csinál? Válasz: D ..... 6 pont
A. 2. Mely értékekre van szükség? Válasz: A, C ..... 6 pont
A. 3. Logikai kifejezés kiértékelése. Válasz: B ..... 6 pont
A. 4. Egyesítés. Válasz: B, D ..... 6 pont
A. 5. Hatványozás. Válasz: A, B, D ..... 6 pont
A. 6. Legnagyobb többszörös. Válasz: C, D ..... 6 pont
A. 7. Adattípus. Válasz: B, C ..... 6 pont
A. 8. Hívások száma. Válasz: C ..... 6 pont
A. 9. Sorozatok. Válasz: A ..... 6 pont
A. 10. Pozitív számok. Válasz: C ..... 6 pont

B RÉSZ ..... **30 pont**

B. 1. Bűvös számok ..... **15 pont**

B.1.a. A `számjegyek(x, b, c)` algoritmus rekurzív változata ..... **5 pont**

– a fejléc tiszteletben tartása ..... 1 pont
– a rekurzív hívás leállási feltétele ..... 1 pont
– újrahívás (logika, paraméterek) ..... 2 pont
– térített értékek ..... 1 pont

```
Algoritmus számjegyek(x, b, c):
   Ha x > 0 akkor
      c[x MOD b] ← 1
      számjegyek(x DIV b, b, c)
   vége(ha)
vége(algoritmus)
```

B.1.b. A `számjegyek(x, b, c)` algoritmus matematikai modellje ..... **3 pont**

$$számjegyek(x,b,c) = \begin{cases} -, & ha\ x = 0 \\ számjegyek(x\ DIV\ b, b, c'), ahol\ c'_{x\ MOD\ b} = 1, & különben \end{cases}$$

B.1.c. Az ***a*** sorozatot felépítő algoritmus ..... **7 pont**

– a *bűvös szám* tulajdonság ellenőrzése
  - V1: a számjegy-halmazok gyakoriságtömbjének azonossága alapján (a ***p***, illetve a ***q*** számrendszerben) ..... 5 pont

```
// felépítjük az x szám számjegyeinek gyakoriságtömbjét a p számrendszerben
// meghatározzuk, rendre, az x szám számjegyeit a q számrendszerben
//     ha az aktuális számjegy nem szerepel a p számrendszerben felírt számban akkor
//        az x szám nem bűvös
//     különben aktualizáljuk a gyakoriság tömb megfelelő elemét
// ha a gyakoriságtömbben maradtak elemek, amelyeknek az értéke 1, az ezeknek
// megfelelő számjegyek megjelennek a p számrendszerben ábrázolt számban és nem
// jelennek meg a q számrendszerben ábrázolt számban
// következik, hogy a szám nem bűvös
bool buvos(int x, int p, int q){
   // eldöntjük, hogy x bűvös-e, tekintve az ábrázolásait a p és q számrendszerben
   int szj[10] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
   szamjegyek(x, p, szj);
   while (x != 0){              // meghatározzuk az x számjegyeit a q számrendszerben
      int uSzj = x % q;
      if (szj[uSzj] == 0)       // ha az aktuális számjegy (a q számrendszerben) nem
         return false;          // jelenik meg a p számrendszerben ábrázolt x-ben
      szj[uSzj]++;
      x = x / q;
   }
   for (int i = 0; i < 10; i++){
      if (szj[i] == 1)          // ha az i számjegyet használtuk a p számrendszerben, és
         return false;          // nem használtuk a q számrendszerben
   }
   return true;
}
```

- V2: más, kisebb hatékonyságú helyes algoritmus .....legtöbb 3 pont

– az ***a*** sorozat felépítése .....2 pont

```
void buvosSzamokSorozata(int p, int q, int n, int &k, int a[]){
   k = 0;
   for (int i = 1; i < n; i++){
      if (buvos(i, p, q))
         a[k++] = i;
   }
}
```

B. 2. Csokik .....15 pont

B.2.a. A körkörös kiszámolást tekinthetjük lineárisnak, több kicsi sorozatban, mindegyikben ***n*** gyerekkel, amely kicsi sorozatokból kialakul egy nagy sorozat ***p*** gyerekkel (***p*** az ***n*** szám többszöröse). A kiszámolás véget ér, amikor az ***n.*** gyerek valamelyik kicsi sorozatban csokit kap (így, a következő gyermek, aki csokit kellene kapjon egy ***k.*** lesz a következő kicsi sorozatban). Például, ha ***n*** = 12 és ***k*** = 9, kialakítunk több kicsi sorozatot, mindegyikben ***n*** gyerekkel:

Sorszámok	1	2	3	4	5	6	7	8	9	10	11	12
Kapott/nem kapott csokit	0	0	0	0	0	0	0	0	0	0	0	0

Sorszámok a nagy sorozatban	1	2	3	4	5	6	7	8	**9**	10	11	12
Gyermek eredeti sorszáma	1	2	3	4	5	6	7	8	**9**	10	11	12
Kapott/nem kapott csokit	0	0	0	0	0	0	0	0	**1**	0	0	0

Sorszámok a nagy sorozatban	13	14	15	16	17	**18**	19	20	21	22	23	24
Gyermek eredeti sorszáma	1	2	3	4	5	**6**	7	8	9	10	11	12
Kapott/nem kapott csokit	0	0	0	0	0	**1**	0	0	1	0	0	0

Sorszámok a nagy sorozatban	25	26	**27**	28	29	30	31	32	33	34	35	**36**
Gyermek eredeti sorszáma	1	2	**3**	4	5	6	7	8	9	10	11	**12**
Kapott/nem kapott csokit	0	0	**1**	0	0	1	0	0	1	0	0	**1**

Itt leáll a csokiosztás, mivel a következő csokit egy olyan gyerek kapná, aki már kapott.

Sorszámok a nagy sorozatban	37	38	39	40	41	42	43	44	45	46	47	48
Gyermek eredeti sorszáma	1	2	3	4	5	6	7	8	9	10	11	12
Kapott/nem kapott csokit	0	0	1	0	0	1	0	0	0	0	0	1

Azoknak a gyerekeknek a sorszáma, akik kapnak csokit, a nagy sorozatban ***k*** többszörösei. .....**2 pont**

B.2.b. Azoknak a gyerekeknek a száma, akik kapnak csokit: ***p*** a ***k*** szám többszöröse. Így, ***p*** = lnko(***n***, ***k***). A ***p*** gyerek közül pontosan ***p*** / ***k*** gyerek kap csokit, tehát azoknak a gyerekeknek a száma, akik nem kapnak csokit: ***n*** - ***p*** / ***k*** = ***n*** - lnko(***n***, ***k***) / ***k*** = ***n*** - (***n*** * ***k*** / lnko(***n***, ***k***)) / ***k*** = ***n*** - ***n*** / lnko(***n***, ***k***).....**3 pont**

B.2.c. Az algoritmus.....**10 pont**

- V1: az ***sz*** szám helyes kiszámítása (a ***sz*** = ***n*** – ***n*** / *lnko*(***n***, ***k***) képlettel).....10 pont

```
// kiszámítja és téríti 2 természetes szám (a és b) lnko-ját
int lnko(int a, int b){
   if ((a == b) && (a == 0))
      return 1;
   if (a * b == 0)
      return a + b;
   while (b != 0){
      int c = b;
      b = a % b;
      a = c;
   }  //while
   return a;
}

int csokik(int n, int k){
   return n - n / lnko(n, k);
}
```

- V2: az ***sz*** érték helyes meghatározása (szimulálás, körkörös lista).....7 pont

```
int csokik(int n, int k){
   const int MAXLEN = 10000000;
   bool csokolade[MAXLEN];
   for (int i = 1; i <= n; i++)
      csokolade[i] = false;
   int i = k;
   int csokiNelkuliGyerekekSzama = n;
   while (!csokolade[i]){ // szimuláljuk a körkörös listát
      csokolade[i] = true;
      csokiNelkuliGyerekekSzama--;
      i += k;
      if (i > n){
         i -= n;
      }
   }
   return csokiNelkuliGyerekekSzama;
}
```