**BABEŞ-BOLYAI TUDOMÁNYEGYETEM**

**MATEMATIKA-INMINDENMATIKA KAR**

**Felvételi verseny (alapképzés) – 2017, július**

**InMindenmatika írásbeli**

**II. TÉTELSOR**

**A versenyzők figyelmébe:**

1. A feladatok megoldásait *pszeudokódban* vagy egy *programozási nyelvben* (*Pascal/C/C*++) kell megadnotok.
2. A megoldások értékelésekor az első szempont az algoritmus ***helyessége***, majd a ***hatékonysága***, ami a *végrehajtási időt* és a *felhasznált memória méretét* illeti.
3. A tulajdonképpeni megoldások előtt, ***kötelezően leírjátok szavakkal az alprogramokat, és megindokoljátok a megoldásotok lépéseit***. Feltétlenül írjatok ***megjegyzéseket*** (kommenteket), amelyek segítik az adott megoldás technikai részleteinek megértését. Írjátok le az azonosítok jelentését és a fölhasznált adatszerkezeteket. Ha ez hiányzik, a tételre kapható pontszámotok 10%-kal csökken.
4. Ne használjatok különleges fejállományokat, előredefiniált függvényeket (például *STL*, karakterláncokat fel­dolgozó sajátos függvények stb.).

**I. Tétel (35 pont)**

1. **Csokik (20 pont)**

Egy reklámcég egy új csokoládét szeretne népszerűsíteni és ebből a célból azt tervezi, hogy kioszt egy-egy csokit ***n*** gyermeknek (10 ≤ ***n*** ≤ 10 000 000), 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 fog csokit kapni (0 < ***k*** < ***n***). Elkezdődik a számolás ***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.*

Írjatok alprogramot, amely kiszámítja azt az ***sz*** számot, amely azoknak a gyermekeknek a száma, akik *nem* kapnak csokit. Bemeneti paraméterek az ***n*** és ***k*** természetes számok, kimeneti paraméter az ***sz*** természetes szám.

**1. *Példa:*** ha ***n*** = 12 és ***k*** = 9, akkor ***sz*** = 8 (nem kap csokit az első, a második, a 4., az 5., a 7., a 8., a 10. és a 11.).

**2. *Példa:*** ha ***n*** = 15 és ***k*** = 7, akkor ***sz*** = 0 (minden gyermek kap csokit).

1. **Megfeleltetés (15 pont)**

Legyen az ***n*** elemű ***a*** és az ***m*** elemű ***b*** sorozat (1 ≤ ***n*** ≤ 10 000 és ***m*** 1 ≤ ***m*** ≤ 10 000), amelyeknek elemei 30 000-nél kisebb természetes számok. Tömbszakaszt alkot egy sorozat egy vagy több eleme, amelyek az eredeti sorozatban egymás utáni pozíciókon találhatók. Az ***a*** sorozat „*megfeleltethető*” a ***b*** sorozatnak, ha az ***a*** sorozatot fel tudjuk osztani diszjunkt tömbszakaszokra úgy, hogy teljesüljenek a következő tulajdonságok:

* ha az összes tömbszakaszt, a felosztás sorrendjében, egymás után ragasztjuk, megkapjuk az ***a*** sorozatot;
* ha az összes tömbszakaszt, a felosztás sorrendjében, behelyettesítjük a megfelelő tömbszakasz elemeinek összegével, megkapjuk, rendre a ***b*** sorozat elemeit.

Írjatok alprogramot, amely eldönti, hogy az ***a*** sorozat *megfeleltethető vagy sem* a ***b*** sorozatnak. Ha igen, találjátok meg azt az elemét a ***b*** sorozatnak (és ennek az elemnek a ***k*** indexét), amely egyenlő az ***a*** sorozat leghosszabb tömb­szakaszához tartozó elemek összegével. Bemeneti paraméterek az ***a*** és ***b*** sorozat, valamint ezeknek a hossza: ***n*** és ***m***. Kimeneti paraméterek a ***válasz***, a ***k*** és a ***maxHossz***, ahol: ***válasz*** értéke ***igaz***, ha a megfeleltetés lehetséges, ellenkező esetben ***hamis***; ***k*** a ***b*** sorozat azon elemének indexe, amely egyenlő a legnagyobb (***maxHossz***) elemszámú tömb­szakasz elemeinek összegével. Ha több ***maxHossz*** elemszámú tömbszakasz létezik, az elsőt vesszük figyelembe. Ha ***válasz*** értéke ***hamis***, ***k*** és ***maxHossz*** értéke -1.

**1. *Példa*:** ha ***n*** = 12, ***a*** = (6, 3, 4, 1, 6, 4, 6, 1, 7, 1, 8, 7), ***m*** = 4 és ***b*** = (13, 7, 18, 16), akkor ***válasz*** = ***igaz***, mivel:

6 + 3 + 4 = 13, 1 + 6 = 7, 4 + 6 + 1 + 7 = 18, 1 + 8 + 7 = 16. Ezek szerint ***k*** = 3 és ***maxHossz*** = 4.

**2. *Példa*:** ha ***n*** = 17, ***a*** = (10, 12, 11, 2, 2, 3, 2, 3, 13, 3, 41, 5, 4, 5, 6, 5, 2), ***m*** = 6 és ***b*** = (33, 4, 15, 41, 25, 2), akkor ***válasz*** = ***hamis***, mivel 10 + 12 + 11 = 33, 2 + 2 = 4, de 3 + 2 + 3 < 15, és 3 + 2 + 3 + 13 > 15. Tehát a ***b*3** = 15 értéket nem tudjuk megfeleltetni az ***a*** sorozat egyik tömbszakaszának sem, mivel nem egyenlő az ***a*** sorozat egymás utáni elemeinek összegével.

*Megjegyzés:* A példában a sorozatot 1-től kezdődően indexeltük.

**II. Tétel (15 pont)**

Legyen a következő alprogram, ahol ***n*** bemeneti paraméter, ***p*** és ***i*** kimeneti paraméterek (***n***, ***p***, ***i*** – természetes számok, 1 ≤ ***n*** ≤ 1 000 000, (0 ≤ ***p*** ≤ 1 000 000, 0 ≤ ***i*** ≤ 1 000 000):

|  |  |
| --- | --- |
| **Algoritmus** f(n, p, i):  **Ha** n ≤ 9 **akkor**  **Ha** n **mod** 2 = 0 **akkor**  { n **mod** 2 *kiszámítja* n *és* 2 *egész osztási maradékát* }  p ← n  i ← 0  **különben**  p ← 0  i ← n  **vége(ha)**  **különben**  f(n **div** 10, p, i)  { n **div** 10 *kiszámítja* n *és* 10 *egész osztási hányadosát* }  **Ha** n **mod** 2 = 0 **akkor**  p ← p \* 10 + n **mod** 10  **különben**  i ← i \* 10 + n **mod** 10  **vége(ha)**  **vége(ha)**  **Vége(algoritmus)** | * 1. Adjátok meg annak a feladatnak a szövegét, amelyet ez az algoritmus old meg.   2. Mi lesz ***p*** és ***i*** értéke az f(205609, p, i) hívás után?   3. Írjátok le egy *iteratív* változatát az adott algoritmusnak, megőrizve a rekurzív   változat fejlécét. |

**III. Tétel (40 pont)**

**Képfeldolgozás**

Egy fehér-fekete képet egy olyan négyzetes tömbbel kódolunk, amelynek elemei nullák (0 = fehér pixel) és egyesek (1 = fekete pixel). A képet a következő műveletekkel alakíthatjuk át:

* Megfordítás (M), vagyis a 0-t 1-gyé, az 1-et 0-ává változtatjuk;
* Forgatás 90 fokkal (F), az óramutató járásával megegyező irányban;
* Zoom (Z), vagyis minden pixel helyére 4 új, az eredetivel azonos értékű pixel kerül.

Egy feldolgozás-sorozatot egy M, F és Z betűkből álló karaktersorozattal írunk le, ahol a betűk sorrendje tetszőleges.

Írjatok programot, amely adott, ***m*** sorral és ***m*** oszloppal rendelkező ***kép*** kétdimenziós tömb (***m*** – természetes szám, 2 ≤ ***m*** ≤ 10) és ***s***, legtöbb öt képfeldolgozó műveletet tartalmazó betűsor esetében végrehajtja ezeket a műveleteket és kiírja az átalakított képet.

***Példa*:** ha ***m*** = 3, ***kép*** = és ***s*** = (F, M, F, Z), akkor a feldolgozás eredménye: .

Az egyes feldolgozások eredményei, rendre:

, , , .

A megoldásban írjatok egy-egy alprogramot, amely:

1. beolvassa a bemeneti adatokata billentyűzetről (ezek garantáltan megfelelnek a követelményeknek);
2. megfordít egy képet;
3. elforgat 90 fokkal egy képet;
4. végrehajtja a zoom műveletet egy képre;
5. kiír a képernyőre egy képet.

**Megjegyzések:**

1. Minden tétel kidolgozása kötelező.
2. A megoldásokat a vizsgalapokra írjátok, (a piszkozatokat nem vesszük figyelembe).
3. Hivatalból jár 10 pont.
4. Rendelkezésetekre áll 3 óra.

**Megoldások**

**I.1. Csokik**

**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 (eredeti sorszáma = 9).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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**

**Algoritmus** lnko(a, b): { *a és b legnagyobb közös osztója* }

**Ha** a = b **és** a = 0 **akkor**

**térítsd** 1

**vége(ha)**

**Ha** a \* b = 0 **akkor**

**térítsd** a + b

**vége(ha)**

**Amíg** b ≠ 0 **végezd el**:

c ← b

b ← a **MOD** b

a ← c

vége(amíg)

**térítsd** a

**Vége(algoritmus)**

**Algoitmus** csokik(n, k):

**térítsd** n - n / lnko(n, k)

**Vége(algoritmus)**

**I.2. Megfeleltetés**

**Algoritmus** megfelel(n, a, m, b, legtöbb, k):

{ *vizsgáljuk a kért tulajdonságot és meghatározzuk a pozíciót a b sorozatban,*

*ahol az az érték található, amelyet elő lehet állítani a legtöbb (a sorozatban található) elem összeadásával* }

legtöbb ← 1 { *a leghosszabb tömbszakasz mérete* }

**Ha** m > n **akkor** { *ha b hosszabb, m a, nem teljesíthető a tulajdonság* }

**térítsd** *hamis*

**vége(ha)**

j ← 1

i ← 1

kezd ← 1 { *az aktuális tömbszakasz kezdő pozíciója* }

**Amíg** j ≤ m **végezd el**: { *a b-nek még nem értünk a végére* }

aktuális ← b[j] { *a b sorozat aktualis eleme* }

s ← 0 { *az aktuális tömbszakasz összege* }

vége ← *hamis* { *még nem találtuk meg a tömbszakaszt* }

**Amíg** **nem** vége **és** i ≤ n **végezd el**: { *az a-nak még nem értünk a végére* }

s ← s + a[i] { *a soron következő elemet a-ból az aktualis összeghez adjuk* }

**Ha** s > aktualis **akkor** { *ha az összeg meghaladja a b sorozat aktuális elemének értékét* }

**térítsd** *hamis* { *nem teljesül a tulajdonság* }

**különben**

**Ha** s = aktuális **akkor** { *ha s-ben megkaptuk a b sorozat aktuális elemének értékét* }

összeadva ← i - kezd + 1 { *meghatározzuk az aktuális tömbszakasz hosszát* }

**Ha** összeadva > legtöbb **akkor** { *aktualizáljuk a leghosszabb tömbszakasz hosszát* }

legtöbb ← összeadva

k ← j

**vége(ha)**

j ← j + 1 { *haladunk b-ben* }

i ← j + 1 { *haladunk a-ban* }

kezd ← i { *a következő tömbszakasz kezdete* }

vége ← *igaz*

**különben**

i ← i + 1

**vége(ha)**

**vége(ha)**

**vége(amíg)**

**Ha** j ≤ m **és** i > n **akkor** { *ha b-nek nincs vége, de a-nak igen* }

**térítsd** *hamis*

**vége(ha)**

**Ha** i ≤ n **és** j > m **akkor** { *ha a-nak nincs vége, de b-nek igen* }

**térítsd** *hamis*

**vége(ha)**

**vége(amíg)**

**térítsd** *igaz*

**Vége(algoritmus)**

**II.**

1. Adjátok meg annak a feladatnak a szövegét, amelyet ez az algoritmus old meg: *Adott az* ***n*** *természetes szám. Építsük fel azt a* ***p*** *számot, amelyet az* ***n*** *szám páros számjegyei alkotnak és azt az* ***i*** *számot, amelyet az* ***n*** *szám páratlan számjegyeiből kaphatunk. A* ***p*** *és* ***i*** *számok számjegyei megőrzik az eredeti sorrendjüket, amelyben előfordultak az* ***n*** *számban.*
2. Mi lesz ***p*** és ***i*** értéke az f(205609, p, i) hívás után? ***p*** = 2060, ***i*** = 59
3. *Iteratív* változat:

**Algoritmus** fIterativ(n, p, i):

p ← 0

i ← 0

páratlanHatv ← 1 { 10 *hatványai, miközben építjük a páratlan számjegyeket tartalmazó számot* }

párosHatv ← 1 { 10 *hatványai, miközben építjük a páros számjegyeket tartalmazó számot* }

**Amíg** n > 0 **végezd el**:

**Ha** n **MOD** 2 ≠ 0 **akkor** { *ha n páratlan* }

i ← i + páratlanHatv \* (n **MOD** 10)

n ← n **DIV** 10

páratlanHatv ← páratlanHatv \* 10

**különben**

p ← p + párosHatv \* (n **MOD** 10)

n ← n **DIV** 10

párosHatv ← párosHatv \* 10

**vége(ha)**

**vége(amíg)**

**Vége(algoritmus)**

**III. Képfeldolgozás**

**Algoritmus** megfordítás(m, kép):

{ *felcseréli (M) egy bináris számjegyeket tároló mátrix szmjegyeit (0-ból 1 lesz és fordítva)* }

**Minden** i = 1, m **végezd el**:

**Minden** j = 1, m **végezd el**:

kép[i][j] ← 1 - kép[i][j]

**vége(minden)**

**vége(minden)**

**Vége(algoritmus)**

**Algoritmus** forgatás(m, kép):{ *elforgat egy kétdimenziós tömböt 90 fokkal (F), az óramutató járásával megegyező irányban* }

**Minden** i = 1, m **végezd el**:

**Minden** j = 1, m **végezd el**:

másolat[i][j] ← kép[i][j]

**vége(minden)**

**vége(minden)**

**Minden** i = 1, m **végezd el**:

**Minden** j = 1, m **végezd el**:

kép[j][m - i + 1] ← másolat[i][j]

**vége(minden)**

**vége(minden)**

**Vége(algoritmus)**

**Algoritmus** zoom(m, kép): { *növeli a mátrix méretét (Z)* }

uj\_i ← 1

**Minden** i = 1, m **végezd el**:

uj\_j ← 1

**Minden** j = 1, m **végezd el**:

ujKép[uj\_i][uj\_j] ← kép[i][j]

ujKép[uj\_i][uj\_j + 1] ← kép[i][j]

ujKép[uj\_i + 1][uj\_j++] ← kép[i][j]

ujKép[uj\_i + 1][uj\_j++] ← kép[i][j]

**vége(minden)**

uj\_i ← uj\_i + 2

**vége(minden)**

m ← m \* 2

**Minden** i = 1, m **végezd el**:

**Minden** j = 1, m **végezd el**:

kép[i][j] ← ujKép[i][j]

**vége(minden)**

**vége(minden)**

**Vége(algoritmus)**

**Algoritmus** kiIr(m, kép): { *kiír egy mátrixot* }

**Minden** i = 1, m **végezd el:**

**Minden** j = 1, m **végezd el:**

**Ki:** kép[i][j], " "

**vége(minden)**

**Ki:** sorvége jel

**vége(minden)**

**Vége(algoritmus)**

**Algoritmus** átalakítás(k, s, m, kép): { *alkalmaz k atalakítást* }

**Minden** i = 1, k **végezd el:**

**Ha** s[i] = 'M' **akkor**

megfordítás(m, kép)

**vége(ha)**

**Ha** s[i] = 'F' **akkor**

forgatás(m, kép)

**vége(ha)**

**Ha** s[i] = 'Z' **akkor**

zoom(m, kép)

**vége(ha)**

**vége(minden)**

**Vége(algoritmus)**