**BABEŞ-BOLYAI TUDOMÁNYEGYETEM**

**MATEMATIKA-INFORMATIKA KAR**

**MATEK-INFO VERSENY – 2016, április**

**INFORMATIKA**

**II. TÉTELSOR**

**A versenyzők figyelmébe:**

1. Adjátok meg az alábbi feladatok megoldásait *pszeudokódban* vagy a *líceumban tanult bármely programozási nyelvben* (*Pascal/C/C*++)!
2. A megoldásokra kapható pontszám első sorban az algoritmusok ***helyességétől*** függ, majd az algorit­musok ***hatékonyságától*** (ami a *végrehajtási időt* és a *felhasznált memória méretét* illeti).
3. Feltétlenül írjatok ***megjegyzéseket*** (kommenteket) amelyek segítik az adott megoldás megértését (írjátok le az azonosítok jelentését és a ***megoldás során alkalmazott ötleteiteket***).
4. Ne használjatok különleges fejállományokban definiált függvényeket (például STL, karakterláncokat feldolgozó sajátos függvények stb.).

**I. Tétel (50 pont)**

|  |  |
| --- | --- |
| 1. **Pascal háromszög (20 pont)**   *A Pascal háromszög* egy olyan egyenlő szárú háromszög, amelynek több, természetes számokat tartalmazó vízszintes sora van: a két egyenlő száron az 1-es számjegy található. Egy adott ***n***. sorban található minden érték a felette levő ***n*** – 1. sor két szomszédos elemének összege, ha ***n*** > 1. A sorokat fentről lefele, 0-val kezdődően számozzuk, ahogy a mellékelt ábrán látható: |  |

Írjatok alprogramot, amely generálja az ***r.*** sor elemeit (2 ≤ ***r*** ≤ 32), anélkül, hogy *kétdimenziós tömböt hasz­nálnátok*. Bemeneti paraméter az ***r*** természetes szám, kimeneti paraméter az ***r***. sor elemeinek sorozata.

1. **Vírusok (10 pont)**

Egy kísérlet során, egy ***n*** létszámú (3 ≤ ***n*** ≤ 1 000) víruspopuláció a következőképpen változik:

1. ha egy bizonyos óra kezdetekor a vírusok létszáma *páros* szám, akkor az óra végére a létszám 50%-kal csökken;
2. ha egy bizonyos óra kezdetekor a vírusok létszáma *páratlan* szám, akkor az óra végére a létszám 1-gyel nő;
3. ha egy bizonyos óra végén, a vírusok létszáma *szigorúan kisebb, mint a túléléshez szükséges kritikus szám*,akkor a vírusok populációja megsemmisül.

Írjatok alprogramot, amely meghatározza hány óra telik el, míg az ***n*** létszámú víruspopuláció megsemmisül. A túléléshez szükséges kritikus számot ***k***-val jelöljük (2 ≤ ***k*** < ***n***), a megsemmisülésig eltelt órák számát ***órákSz*** jelöli. Az alprogram bemeneti paraméterei ***n*** és ***k***, a kimeneti paramétere ***órákSz*** lesz***.***

***Példa:*** ha ***n*** = 11 és ***k*** = 3, a populáció ***órákSz*** = 5 óra alatt semmisül meg.

1. **Maximális szorzat (20 pont)**

Adott az ***n*** elemű (3 ≤ ***n*** ≤ 10 000), egész számokat tároló ***x*** sorozat, ahol az elemek értéke nagyobb, mint -30 000 és kisebb, mint 30 000.

Írjatok alprogramot, amely meghatároz *három* számot az ***x*** sorozatból, amelyeknek a szorzata *maximális*. Az alprogram bemeneti paraméterei ***n*** és ***x***, kimeneti paraméterei az ***a***, ***b*** és ***c*** számok, amelyek az ***x*** sorozat elemei és rendelkeznek a kért tulajdonsággal. Ha a feladatnak több megoldása van, csak egyet kell megadnotok.

***Példa:*** ha ***n*** = 10 és ***a*** = (3, -5, 0, 5, 2, -1, 0, 1, 6, 8), a három szám: ***a*** = 5, ***b*** = 6, ***c*** = 8.

**II. Tétel (15 pont)**

Legyen a következő alprogram, ahol az ***a*** bemeneti paraméter természetes szám (0 < ***a*** ≤ 30 000):

**Algoritmus** F(a)**:**

b ← 0

p ← 1

**Amíg** a > 0 **végezd el:**

c **←** a **mod** 10 { **mod** *kiszámítja az* **a** *szám* **10***-zel való egész osztási maradékát* }

**Ha** c **mod** 2 ≠ **0 akkor**

b **←** b + p \* c

p **←** p \* 10

**vége(ha)**

a **←** a **div** 10 { **div** *kiszámítja az* **a** *szám* **10***-zel való egész osztási hányadosát* }

**vége(amíg)**

**térítsd b**

**Vége(algoritmus)**

**a.** Adjátok meg annak a feladatnak a szövegét, amelyet ez az algoritmus old meg.

**b.** Mit térít az **F(2103)** hívás?

**c.** Írjátok le az adott algoritmus *rekurzív* változatát, amelynek fejléce azonos az iteratív algoritmus fejlécével.

**III. Tétel (25 pont)**

Egy természetes számokat tartalmazó sorozatot *palindrom*nak nevezünk, ha balról jobbra olvasva, ugyanazt a sorozatot kapjuk, mintha jobbról balra haladva olvasnánk. Például az (1, 2, 3, 2, 1) sorozat *palindrom*, míg az (1, 2, 3, 2, 4) nem *palindrom*. Egy természetes számokat tartalmazó sorozat *ciklikus palindrom*, ha az elemeinek néhány körkörös permutációjával *palindrommá* alakítható. Az elemek körkörös permutációja alatt a sorozat elemeinek egy pozícióval balra tolását értjük (kivételt képez az első elem, amely a sorozat utolsó pozíciójára kerül).

Írjatok programot, amely eldönti, hogy az ***n*** elemű (1 ≤ ***n*** ≤ 1 000), természetes számokat tartalmazó ***a*** sorozat *ciklikus palindrom-e vagy sem*, és kiír egy megfelelő üzenetet (*Igen*/*Nem*). Ha a döntés eredménye *Igen*, a program meghatározza azoknak a körkörös permutációknak a számát, amelyekkel a sorozat palindrommá alakítható.

***Példák:***

* az ***a*** = (1, 1, 2, 2) sorozat az (1, 2, 2, 1) egyetlen körkörös permutációval palindrommá alakítható.
* az ***a*** = (3, 4, 3, 2, 1, 1, 2) sorozat öt körkörös permutációval palindrommá alakítható:

(4, 3, 2, 1, 1, 2, 3); (3, 2, 1, 1, 2, 3, 4); (2, 1, 1, 2, 3, 4, 3); (1, 1, 2, 3, 4, 3, 2); (1, 2, 3, 4, 3, 2, 1).

* az ***a*** = (1, 2, 3) sorozat nem alakítható palindrommá körkörös permutációkkal.

Írjatok egy-egy alprogramot, amely:

1. beolvassa az ***a*** sorozatota billentyűzetről;
2. kiírja az*Igen/Nem*üzenetet a képernyőre és *Igen* esetében az elvégzendő körkörös permutációk számát;
3. eldönti egy sorozatról, hogy ciklikus palindrom-e vagy sem;
4. meghatározza a szükséges körkörös permutációk számát.

**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.

**Megoldás**

**1: Pascal háromszög**

A sorozat és az újSorozat maximális hossza = 33. A sorozat az eredményt tárolja.

Algoritmus PascalHáromszög(r, sorozat):

{ *kiszámítjuk a Pascal háromszög egy-egy sorának elemeit az előtte levő sor értékeinek alapján* }

sorozat[0] **←** 1

sorozat[1] **←** 1 { *a háromszög 2. sora* }

**Minden** k **←** 2, r **végezd el:**

újSorozat[0] **←** 1

újSorozat[k] **←** 1

**Minden** i = 1, k – 1 **végezd el**: { *az új sorozatot az előtte levő alapján számítjuk ki* }

újSorozat[i] **←** sorozat[i - 1] + sorozat[i]

**vége(minden)**

**Minden** i = 0, k **végezd el:** { *az új sorozat elfoglalja a régi helyét* }

sorozat[i] **←** újSorozat[i]

**vége(minden)**

**vége(minden)**

**Vége(algoritmus)**

**2: Vírusok**

Meghatározza hány óra telik el, míg az ***n*** létszámú víruspopuláció megsemmisül. A túléléshez szükséges kritikus számot ***k***-val jelöljük (2 ≤ ***k*** < ***n***), a megsemmisülésig eltelt órák számát ***órákSz***.

**Algoritmus** vírusok(n, k):

{ *követjük a populáció változását* }

megsemmisült **←** n < k

órákSz **←** 0

**Amíg** **nem** megsemmisült **végezd el**:

**Ha** n **MOD** 2 = 0 **akkor** { *páros* *szám-e?* }

n **←** n **DIV** 2 { *ha a vírusok száma páros, felezzük a populációt* }

**különben**

n **←** n + 1 { *ha a vírusok száma páratlan, növeljük eggyel a populációt* }

**vége(ha)**

órákSz **←** órákSz + 1

megsemmisült **←** n < k { *vizsgáljuk, hogy a populáció megsemmisült-e* }

**vége(amíg)**

**térítsd** órákSz

**Vége(algoritmus)**

**3. Maximális szorzat**

**Lásd a december 15.-én tartott felkészítő anyagát.**

**II.**

**a.** A feladat szövege, amelyet az adott algoritmus old meg: „Határozzátok meg azt a számot, amelyet az adott ***a*** természetes szám páratlan értékű számjegyei alkotnak abban a sorrendben, amelyben az ***a***-ban előfordul­tak. Ha ***a***-nak nincs páratlan számjegye, az eredmény 0.

**b.** Mit térít az F(2103) hívás? 13-at.

**c.** Az adott algoritmus *rekurzív* változata, amelynek fejléce azonos az iteratív algoritmus fejlécével.

**Algoritmus** rekurzívFüggvény(a):

**Ha** a < 1 **akkor**

**térítsd** a

**különben**

c **←** a **MOD** 10

**Ha** c **MOD** 2 ≠ 0 **akkor**

**térítsd** c + 10 \* rekurzívFüggvény(a **DIV** 10)

**különben**

**térítsd** rekurzívFüggvény(a **DIV** 10)

**vége(ha)**

**vége(ha)**

**Vége(algoritmus)**

**III: Körkörös palindrom (ciklikus)**

**Algoritmus** palindrom(a, start, vége): { *vizsgáljuk, hogy a sorozat* [*start..vége*] *tömbszakasza palindrom-e* }

i ← start

j ← vége

{ *páronként összehasonlítjuk az elemeket (az indexek szimmetrikusak a tömbszakasz közepétől tekintve)* }

**Amíg** a[i] = a[j] és i < j **végezd el**:

i ← i + 1

j ← j - 1

**vége(amíg)**

**térítsd** i ≥ j

**Vége(algoritmus)**

**Algoritmus** permutálásokSzáma(n, a):

{ *meghatározzuk azoknak a körkörös permutálásoknak a számát, amelyek az n elemű a sorozatot palindrommá*

*alakítják; ha a sorozat nem alakítható palindrommá ezekkel a körkörös permutációkkal, -1-et térítünk* }

**Minden** i = 1, n **végezd el**: { *kétszer egymás után másoljuk a sorozatot* }

a[n + i] ← a[i]

megvan ← *hamis*

i ← 0

**Amíg nem** megvan **és** i < n **végezd el**:

i ← i + 1 { *ellenőrizzük, hogy az i. helyen kezdődő és n elemű tömbszakasz palindrom-e* }

megvan ← palindrom(a, i, i + n - 1)

**vége(amíg)**

**Ha** megvan **akkor**

**térítsd** i - 1

**különben**

**térítsd** -1;

**Vége(algoritmus)**