**Felvételi előkészítő**

**(Rekurzió)**

1. **Ackermann**

Legyenek az ***m*** és ***n*** természetes számok (0 ≤ ***m*** ≤ 10, 0 ≤ ***n*** ≤ 10), valamint az Ack(m, n) algoritmus, amely kiszámítja az *Ackermann*-függvény értékét ***m*** és ***n*** esetében. Állapítsátok meg, hányszor hívja meg önmagát az Ack(m, n) algoritmus a következő utasítások végrehajtásának következtében.

|  |
| --- |
| m ← 1, n ← 2  Ack(m, n) |

|  |  |  |
| --- | --- | --- |
| **Algoritmus** Ack(m, n)  **Ha** m = 0 **akkor**  **térítsd** n + 1  **különben**  **Ha** m > 0 **és** n = 0 **akkor**  **térítsd** Ack(m - 1, 1)  **különben**  **térítsd** Ack(m - 1, Ack(m, n - 1))  **vége(ha)**  **vége(ha)**  **Vége(algoritmus)** | 1. 7-szer 2. 10-szer 3. 5-ször 4. ugyanannyiszor, mint az alábbi utasítások végrehajtásá­nak következtében:  |  | | --- | | m ← 1, n ← 3  Ack(m, n) | |

1. **Mely értékek szükségesek?**

Legyen a különbség(a, n) algoritmus, ahol ***a*** egy ***n*** elemű (0 < ***n*** < 100) sorozat, amely egész számokat tárol:

|  |
| --- |
| **Algoritmus** különbség(a, n)  **Ha** n = 0 **akkor** **térítsd** 0  **vége(ha)**  **Ha** |a[n]| **MOD** 2 = 0 **akkor** { |a[n]| *az* a[n] *szám abszolút értékét jelöli* }  **térítsd** különbség(a, n - 1) + a[n]  **különben**  **térítsd** különbség(a, n - 1) - a[n]  **vége(ha)**  **Vége(algoritmus)** |

Az ***n*** és ***a*** mely értékeire térít a fenti algoritmus 0-át?

|  |  |
| --- | --- |
| 1. ***n*** = 4 és ***a*** = (6, 4, 5, 5) | 1. ***n*** = 4 és ***a*** = (-6, 5, 4, -7) |
| 1. ***n*** = 8 és ***a*** = (-6, 5, -1, -4, 1, 4, -7, 6) | 1. ***n*** = 8 és ***a*** = (-6, -3, 0, 1, 2, 3, -1, 4) |

1. **Kiegészítés (6 pont)**

Legyen a kizárPáratlan(n) algoritmus, ahol ***n*** (1 ≤ ***n*** ≤ 100 000) természetes szám. Állapítsátok meg, melyik utasítást kellene a „…” helyére írni, ahhoz, hogy az algoritmus zárja ki az ***n*** számból a páratlan értékű számjegyeket.

|  |
| --- |
| **Algoritmus** kizárPáratlan(n)  **Ha** n = 0 **akkor**  **térítsd** 0  **vége(ha)**  **Ha** n **MOD** 2 = 1 **akkor**  **térítsd** kizárPáratlan(n **DIV** 10)  **vége(ha)**  **térítsd** ...  **Vége(algoritmus)** |

1. kizárPáratlan(n **MOD** 10) \* 10 + n **DIV** 10
2. kizárPáratlan(n) \* 10 + n **MOD** 10
3. kizárPáratlan(n **DIV** 10) \* 10 + n **MOD** 10
4. kizárPáratlan((n **DIV** 10) **MOD** 10) \* 10
5. **Számjegyszorzat**

A számJegyek(n, d) algoritmus (***n*** és ***d*** természetes számok, 10 ≤ ***n*** ≤ 100 000, 1 ≤ ***d*** ≤ 9), meghatározza és visszatéríti azt a legkisebb természetes számot, amelynek ***d***-nél kisebb vagy ***d***-vel egyenlő, nem nulla szám­jegyei vannak és amely számjegyeknek a szorzata egyenlő ***n***-nel. Például, ha ***n*** = 108 és ***d*** = 9, az algoritmus 269-et térít vissza. Ha ilyen szám nem létezik, az algoritmus -1-et térít. Állapítsátok meg, hányszor hívja meg önmagát a számJegyek(n, d) algoritmus az alábbi programrészlet végrehajtásának következtében:

|  |  |  |
| --- | --- | --- |
| **Algoritmus** számJegyek(n, d)  **Ha** d = 1 **akkor**  **Ha** n = 1 **akkor**  **térítsd** 0  **különben**  **térítsd** -1  **vége(ha)**  **különben**  **Ha** n **MOD** d = 0 **akkor**  érték ← számJegyek(n **DIV** d, d)  **Ha** érték < 0 **akkor**  **térítsd** -1  **különben**  **térítsd** érték \* 10 + d  **vége(ha)**  **különben**  **térítsd** számJegyek(n, d - 1)  **vége(ha)**  **vége(ha)**  **Vége(algoritmus)** | |  | | --- | | beOlvas n  érték ← számJegyek(n, 9) |  1. Ha ***n*** = 108, az algoritmus 11-szer hívja meg önmagát. 2. Ha ***n*** = 109, az algoritmus 8-szor hívja meg önmagát. 3. Ha ***n*** = 13, az algoritmus egyszer sem hívja meg önmagát. 4. Ha ***n*** = 100, az algoritmus 10-szer hívja meg önmagát. |

1. **Varázslat**

Egy számjegymágus olyan varázslatot végez, amelynek eredményeképpen egy ***x*** természetes szám (100 < ***x***< 1 000 000, amelynek a 10-es számrendszerben van legkevesebb két 0-tól különböző számjegye) szétválik két pozitív természetes számra: a ***bal*** és ***jobb*** számokra, amelyek egymás után ragasztva megadják az ***x*** számot. Ugyanakkor a ***bal*** és ***jobb*** számok szorzata a lehető legnagyobb. Például, ha ***x*** = 1 092, a varázslat szétválasztja a ***bal*** = 10 és ***jobb*** = 92 számokra.

Az alábbi algoritmusok közül melyik alkalmazza a varázslatot az ***x*** természetes számra, amelynek 10-es számrendszerben van legkevesebb két 0-tól különböző számjegye (100 ≤ ***x*** ≤ 1 000 000)? Az algoritmus meghatározza a ***z*** természetes számban (0 ≤ ***z*** ≤ 1 000 000) az ***x*** szám ***jobb*** részét. Az alábbi algoritmusok léteznek:

* hatvány(b, p) – meghatározza a ***bp*** értéket, ***b***, ***p***– természetes számok (1 ≤ ***b*** ≤ 20, 1 ≤ ***p*** ≤ 20);
* szjSzáma(sz) – meghatározza az ***sz*** szám (0 ≤ ***sz*** ≤ 1 000 000) számjegyeinek darabszámát;

|  |  |
| --- | --- |
|  | **Algoritmus** varázslat(x, z)  maxSzorzat ← -1  eredmény ← 0  **Amíg** x > 0 **végezd el**  z ← (x **MOD** 10) \* hatvány(10, szjSzáma(z)) + z  x ← x **DIV** 10  **Ha** x \* z > maxSzorzat **akkor**  maxSzorzat ← x \* z  eredmény ← z  **vége(ha)**  **vége(amíg)**  **térítsd** maxSzorzat  **Vége(algoritmus)** |
| 1. a | **Algoritmus varázslat**(x, z)  t ← 0  **Ha** x > 0 **akkor**  y ← (x **MOD** 10) \* hatvány(10, szjSzáma(z)) + z  t ← x **DIV** 10  **Ha** x \* z < y \* t **akkor**  **térítsd** varázslat(y, t)  **különben**  **térítsd** t  **vége(ha)**  **különben**  **térítsd** t  **vége(ha)**  **Vége(algoritmus)** |
|  | **Algoritmus varázslat**(x, z)  maxSzorzat ← -1  eredmény ← 0  **Amíg** x > 0 **végezd el**  z ← (x **MOD** 10) \* hatvány(10, szjSzáma(z)) + z  x ← x **DIV** 10  **Ha** x \* z > maxSzorzat **akkor**  maxSzorzat ← x \* z  eredmény ← z  **vége(ha)**  **vége(amíg)**  **térítsd** eredmény  **Vége(algoritmus)** |
|  | **Algoritmus varázslat**(x, z)  **Ha** x > 0 **akkor**  y ← (x **MOD** 10) \* hatvány(10, szjSzáma(z)) + z  t ← x **DIV** 10  **Ha** x \* z < y \* t **akkor**  **térítsd** varázslat(y, t)  **különben**  **térítsd** z  **vége(ha)**  **különben**  **térítsd** z  **vége(ha)**  **Vége(algoritmus)** |

1. **Számolás**

Legyen a számol(a, b) algoritmus, amelynek bemeneti paraméterei az ***a*** és ***b*** pozitív természetes számok, ahol 1 ≤ ***a*** ≤ 100, 1 ≤ ***b*** ≤ 100.

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6. | **Algoritmus** számol(a, b):  **Ha** a ≠ 0 **akkor**  **térítsd** számol(a **DIV** 2, b + b) + b \* (a **MOD** 2)  **vége(ha)**  **térítsd** 0  **Vége(algoritmus)** |

Az alábbi válaszok közül melyek hamisak?

1. ha ***a*** és ***b*** egyenlők, az algoritmus ***a*** értékét téríti
2. ha ***a*** = 1000 és ***b*** = 2, az algoritmus 10-szer hívja meg önmagát
3. az algoritmus által kiszámított és térített érték egyenlő (***a*** / 2 + 2 \* ***b***)-vel
4. az 5. sorban található utasítás egyszer sem hajtódik végre
5. az 5. sorban található utasítás egyszer hajtódik végre
6. **Törzstényezők**

|  |  |
| --- | --- |
| Legyen a törzsTényezők(n, d, k, x) algoritmus, amely meghatározza az ***n*** természetes szám ***k*** darab törzstényezőjét, a törzstényezők keresését a ***d*** értéktől kezdve. Bemeneti para­mé­terek az ***n***, ***d*** és ***k*** számok, kimeneti paraméterek az ***x*** sorozat, amely a ***k*** törzstényezőt tartalmazza (1 ≤ ***n*** ≤ 10000, 2 ≤ ***d*** ≤ 10000, 0 ≤ ***k*** ≤ 10000).  Határozzátok meg, hányszor hívja meg önmagát a törzs­Tényezők(n, d, k, x) algoritmus a következő programrészlet végrehajtásának következtében: | **Algoritmus** törzsTényezők(n, d, k, x):  **Ha** n **MOD** d = 0 **akkor**  k ← k + 1  x[k] ← d  **vége(ha)**  **Amíg** n **MOD** d = 0 **végezd el**  n ← n **DIV** d  **vége(amíg)**  **Ha** n > 1 **akkor**  törzsTényezők(n, d + 1, k, x)  **vége(ha)**  **Vége(algoritmus)** |

|  |
| --- |
| n ← 120  d ← 2  k ← 0  törzsTényezők(n, d, k, x) |

|  |  |  |  |
| --- | --- | --- | --- |
| **A.** 3-szor | **B.** 5-ször | **C.** 9-szer | **D.** 6-szor |

**E.** ugyanannyiszor, mint a következő programrészlet esetében:

|  |
| --- |
| n ← 750  d ← 2  k ← 0  törzsTényezők(n, d, k, x) |

1. **Konverzió**

Legyen a konverzió(s, hossz) algoritmus, amely átalakítja az ***s*** karakterláncot, amely egy 16-os számrendszerben ábrázolt szám, a megfelelő 10-es számrendszerben érvényes alakjára. Az ***s*** karakterlánc ***hossz*** darab karaktert tartalmaz, ahol a karakter értéke lehet egy '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' számjegy, vagy egy 'A', 'B', 'C', 'D', 'E', 'F' nagybetű (a ***hossz*** természetes szám, 1 ≤ ***hossz*** ≤ 10).

Írjátok le a konverzió(s, hossz) algoritmus *rekurzív* változatát úgy, hogy a fejléce és a hatása legyen azonos az alábbi algoritmuséval:

|  |
| --- |
| **Algoritmus** konverzió(s, hossz):  szám ← 0  **Minden** i = 1, hossz **végezd el**  **Ha** s[i]≥ 'A' **akkor**  szám ← szám \* 16 + s[i] - 'A' + 10  **különben**  szám ← szám \* 16 + s[i] - '0'  **vége(ha)**  **vége(minden)**  **térítsd** szám  **Vége(algoritmus)** |

1. **Vajon mit csinál? (5p)**

Adott a kifejezés(n) algoritmus, ahol ***n*** (1 ≤ ***n*** ≤ 10000) természetes szám.

|  |
| --- |
| **Algoritmus** kifejezés(n):  **Ha** n > 0 **akkor**  **Ha** n **MOD** 2 = 0 **akkor**  **térítsd** -n \* (n + 1) + kifejezés(n - 1)  **különben**  **térítsd** n \* (n + 1) + kifejezés(n - 1)  **vége(ha)**  **különben**  **térítsd** 0  **vége(ha)**  **Vége(algoritmus)** |

Állapítsátok meg az *E*(*n*) kifejezésnek azt a matematikai alakját, amelyet a fenti algoritmus számít ki:

* 1. E(*n*) = 1\* 2 - 2 \* 3 + 3 \* 4 + - 4 \* 5... + (-1)*n*+1 \* *n* \* (*n* + 1)
  2. E(*n*) = 1\* 2 - 2 \* 3 + 3 \* 4 + ... + (-1)*n* \* *n* \* (*n* + 1)
  3. E(*n*) = 1\* 2 + 2 \* 3 + 3 \* 4 + ... + (-1)*n*+1 \* *n* \* (*n* + 1)
  4. E(*n*) = 1\* 2 + 2 \* 3 + 3 \* 4 + ... + (-1)*n* \* *n* \* (*n* + 1)
  5. E(*n*) = 1\* 2 - 2 \* 3 - 3 \* 4 - ... - (-1)*n* \* *n* \* (*n* + 1)

1. **Tegyél 'b' betűket**

Legyen az ***n*** × ***n*** méretű négyzetes ***mat*** tömb (***n*** – páratlan természetes szám, 3 ≤ ***n*** ≤ 100). A tegyélB(mat, n, i, j) algoritmus 'b' betűket tesz a ***mat*** tömb bizonyos pozícióira. Az ***i*** és ***j*** paraméterek természetes számok (1 ≤ ***i*** ≤ ***n***, 1 ≤ ***j*** ≤ ***n***).

|  |  |  |
| --- | --- | --- |
| **Algoritmus** tegyélB(mat, n, i, j):  **Ha** i ≤ n **DIV** 2 **akkor**  **Ha** j ≤ n - i **akkor**  mat[i][j] ← 'b'  tegyélB(mat, n, i, j + 1)  **különben**  tegyélB(mat, n, i + 1, i + 2)  **vége(ha)**  **vége(ha)**  **Vége(algoritmus)** |  | Határozzátok meg, hányszor hívja meg önmagát a tegyélB(mat, n, i, j) algoritmus a következő programrészlet vég­rehajtásának következtében: |
|  | n ← 7  i ← 2  j ← 4  tegyélB(mat, n, i, j) |

|  |  |  |  |
| --- | --- | --- | --- |
| **A.** 5-ször | **C.** 10-szer | **D.** 0-szor | **E.** végtelenszer |

|  |  |
| --- | --- |
| **B.** ugyanannyiszor, mint a mellékelt programrészlet esetében: | n ← 9, i ← 3, j ← 5  tegyélB(mat, n, i, j) |

1. **Számolás - karakterekkel**

Legyen a számolásKarakterekkel(s, n, p, q, szám) algoritmus, ahol ***s*** egy ***n*** karakterből álló sorozat (***n*** természetes szám, 1 ≤ ***n*** ≤ 9), ***p***, ***q*** és ***szám*** természetes számok (1 ≤ ***p*** ≤ ***n***, 1 ≤ ***q*** ≤ ***n***, ***p*** ≤ ***q***).

|  |
| --- |
| **Algoritmus** számolásKarakterekkel(s, n, p, q, szám):  eredmény ← 0  i ← p  **Amíg** i ≤ q **végezd el**  **Amíg** i ≤ q **és** s[i] ≥ '0' **és** s[i] ≤ '9' **végezd el**  szám ← szám \* 10 + s[i] - '0'  i ← i + 1  **vége(amíg)**  eredmény ← eredmény + szám  szám ← 0  i ← i + 1  **vége(amíg)**  **térítsd** eredmény  **Vége(algoritmus)** |

Írjátok le a számolásKarakterekkel(s, n, p, q, szám) algoritmus *rekurzív* változatát úgy, hogy a fejléce és a hatása legyen azonos a fenti algoritmuséval. Az alábbi programrészletből hívjuk meg:

|  |
| --- |
| **Beolvas:** n, s, p, q  **KiÍr:** számolásKarakterekkel(s, n, p, q, 0) |

1. **Logikai kifejezés kiértékelése**

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

|  |
| --- |
| **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 program­részlet végrehajtásának következtében:

|  |
| --- |
| s ← (*hamis*, *hamis*, *hamis*, *hamis*, *hamis*, *hamis*, *igaz*, *hamis*, *hamis*, *hamis*)  k ← 10  i ← 3  kiértékelés(s, k, i) |

1. 3-szor
2. ugyanannyiszor, mint a következő programrészlet esetében

|  |
| --- |
| s ← (*hamis*, *hamis*, *hamis*, *hamis*, *hamis*, *hamis*, *hamis*, *igaz*)  k ← 8  i ← 4  kiértékelés(s, k, i) |

1. 6-szor
2. egyszer sem
3. végtelenszer
4. **Egyesítés**

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 ≤ ***n*** ≤ 200, 1 ≤ ***x*** ≤ 1000).

Legyen az alább megadott egyesítés(a, n, b, m, c, p) algoritmus, ahol ***a***, ***b*** és ***c*** sorozatok, amelyek természetes számokat tároló és rendre ***n***, ***m*** és ***p*** elemű halmazokat ábrázolnak (1 ≤ ***n*** ≤ 200, 1 ≤ ***m*** ≤ 200, 1 ≤ ***p*** ≤ 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 5. c[p] ← b[i] 6. **vége(minden)** 7. **különben** 8. **Ha nem** eleme(a[n], b, m) **akkor** 9. p ← p + 1 10. c[p] ← a[n] 11. **vége(ha)** 12. egyesítés(a, n - 1, b, m, c, p) 13. **vége(ha)** 14. **Vége(algoritmus)** |

A következő állítások közül melyek bizonyulnak mindig igaznak?

1. ha az ***a*** halmaz egy elemet tartalmaz, az egyesítés(a, n, b, m, c, p) algoritmus végtelen ciklusba kerül
2. ha az ***a*** halmaznak négy eleme van, az egyesítés(a, n, b, m, c, p) algoritmus első meghívása maga után vonja a 12. sorban található utasítás végrehajtását négyszer
3. ha az ***a*** halmaznak öt eleme van, az egyesítés(a, n, b, m, c, p) algoritmus első meghívása maga után vonja a második sorban található utasítás végrehajtását ötször
4. ha az ***a*** halmaznak ugyanannyi eleme van, mint a ***b*** halmaznak, az egyesítés(a, n, b, m, c, p) algoritmus végrehajtása után a ***c*** halmaznak ugyanannyi eleme lesz, mint az ***a*** halmaznak
5. ha az ***a*** és ***b*** halmazok elemei azonosak, az egyesítés(a, n, b, m, c, p) algoritmus végrehajtása után a ***c*** halmaznak ugyanannyi eleme lesz, mint az ***a*** halmaznak
6. **Hatványra emelés**

Melyik algoritmus számítja ki helyesen ***ab*** értékét, ahol ***a*** és ***b*** természetes számok (1 ≤ ***a*** ≤ 11, 0 ≤ ***b*** ≤ 11)?

|  |  |  |  |
| --- | --- | --- | --- |
|  | **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)** |  | **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)** |
|  | **Algoritmus** hatvány(a, b):  eredmény ← 1  **Amíg** b > 0 **végezd el**  eredmény ← eredmény \* a  b ← b - 1  **vége(amíg)**  **térítsd** eredmény  **Vége(algoritmus)** |  | **Algoritmus** hatvány(a, b):  **Ha** b = 0 **akkor**  **térítsd** 1  **vége(ha)**  aux ← hatvány(a, b **DIV** 2)  **Ha** b **MOD** 2 = 0 **akkor**  **térítsd** aux \* aux  **különben**  **térítsd** a \* aux \* aux  **vége(ha)**  **Vége(algoritmus)** |
|  | **Algoritmus** hatvány(a, b):  **Ha** b = 0 **akkor**  **térítsd** 1  **vége(ha)**  **térítsd** a \* hatvány(a, b - 1)  **Vége(algoritmus)** |  |  |

1. **Polinom értéke**

Adott a kiértékelés(n, egyh, x) algoritmus, ahol ***egyh*** egy ***n* + 1** elemű, valós számokat tároló sorozat, amelynek értékei a [-100, 100] intervallumhoz tartoznak és amelyek az ***n*** fokú ***P*(*x*)** = ***egyh*1 \* *xn* + *egyh*2 \* *xn* - 1 + .... + *egyhn* \* *x* + *egyhn*+ 1** polinom együtthatói, ***x*** csökkenő hatványainak sorrendjében (***n*** természetes szám, 1 ≤ ***n*** ≤ 10). Az algoritmus meghatározza a polinom értékét egy adott ***x*** pontban (***x*** valós szám, amely a [-10, 10] intervallumhoz tartozik.

|  |  |
| --- | --- |
| **Algoritmus** kiértékelés(n, egyh, x):  érték ← 0.0  **Minden** i ← 1, n + 1 **végezd el**  érték ← érték \* x + egyh[i]  **vége(minden)**  **térítsd** érték  **Vége(algoritmus)** | Írjátok le a kiértékelés(n, egyh, x) algoritmus *rekurzív* válto­zatát úgy, hogy a fejléce és a hatása legyen azonos a fenti algorit­muséval. |

|  |  |
| --- | --- |
| **Algoritmus** f(a, b, sz):  k ← 0  **Amíg** b < sz **végezd el:**  k ← k + 1  b ← a + b  a ← b - a  **vége(amíg)**  **térítsd** k  **Vége(algoritmus)** | Adott a következő algoritmus, amelynek három, nem nulla természetes szám bemeneti paramétere van: ***a****,* ***b*** és ***sz***, amelyeknek értékei kisebbek, mint 10 000:   * 1. Adjátok meg a feladat szövegét, amelyet ez az algoritmus old meg, ha ***a*** = 1 és ***b*** = 0 értékekre hívjuk meg.   2. Mit térít az f(1,0,10) hívás?   3. Írjátok le a fenti algoritmusnak egy rekurzív változatát, megőrizve az iteratív (nem rekurzív) változat fejlécét. |

|  |  |
| --- | --- |
| **Algoritmus** f(n, p, i):  **Ha** n ≤ 9 **akkor**  **Ha** n **mod** 2 = 0 **akkor**  p ← n  i ← 0  **különben**  p ← 0  i ← n  **vége(ha)**  **különben**  f(n **div** 10, p, i)  **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)** | 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):   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. |

|  |  |
| --- | --- |
| **Algoritmus** F(a)**:**  b ← 0  p ← 1  **Amíg** a > 0 **végezd el:**  c **←** a **mod** 10  **Ha** c **mod** 2 ≠ 0 **akkor**  b **←** b + p \* c  p **←** p \* 10  **vége(ha)**  a **←** a **div** 10  **vége(amíg)**  **térítsd** b  **Vége(algoritmus)** | Legyen a következő alprogram, ahol az ***a*** bemeneti paraméter természe­tes szám (0 < ***a*** ≤ 30 000):  **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. |

|  |  |
| --- | --- |
| **Algoritmus** F(a, b):  c **←** 1  **Amíg** b > 0 **végezd el:**  **Ha** b **mod** 2 = 1 **akkor**  c **←** (c \* a) **mod** 10  **vége(ha)**  a **←** (a \* a) **mod** 10  b **←** b **div** 2  **vége(amíg)**  **térít** c  **Vége(algoritmus)** | Adott a következő alprogram, ahol az ***a*** és ***b*** (0 < ***a*** ≤ 10 000, 0 ≤ ***b*** ≤ 10 000) természetes számok bemeneti paraméterek.   * 1. Adjátok meg annak a feladatnak a szövegét, amelyet ez az algoritmus old meg.   2. Mit térít az F(1002,6) hívás?   3. Írjátok le egy *rekurzív* változatát a fenti iteratív (nem rekurzív) algoritmusnak. A fejléce legyen azonos a fenti algoritmus fejlécével. |

**Kitűzött feladatok**

1. **Hatvány**

A következő rekurzív algoritmus minimális számú szorzással számítja ki az ***xn*** hatvány értékét (***x*** – valós szám, 0 < ***x*** ≤ 10, ***n*** – természetes szám, 1 ≤ ***n*** ≤ 40).

Állapítsátok meg, hány szorzást végez el az algoritmus, ha ***x*** és ***n*** alább megadott értékeire hívjuk meg.

|  |  |
| --- | --- |
| **Algoritmus** f(x, n)  **Ha** n = 0 **akkor**  **térítsd** 1  **különben**  **Ha** n *páratlan szám* **akkor**  **térítsd** f(x \* x, n **DIV** 2) \* x  **különben**  **térítsd** f(x \* x, n **DIV** 2)  **vége(ha)**  **vége(ha)**  **Vége(algoritmus)** | 1. Ha ***x*** = 1.5 és ***n*** = 10, akkor a szorzások száma= 6 2. Ha ***x*** = 2.55 és ***n*** = 1, akkor a szorzások száma= 1 3. Ha ***x*** = 3.14 és ***n*** = 32, akkor a szorzások száma = 7 4. Ha ***x*** = 0.5 és ***n*** = 1, akkor a szorzások száma= 2 |

1. **Számolás – karakterekkel**

Legyen a számolásKarakterekkel(s, n, i, szám) algoritmus, ahol ***s*** egy ***n*** karakterből álló sorozat (***n*** természetes szám, 1 ≤ ***n*** ≤ 200), ***i*** és ***szám*** természetes számok (1 ≤ ***i*** ≤ ***n***).

|  |
| --- |
| **Algoritmus** számolásKarakterekkel(s, n, i, szám):  eredmény ← 0  **Amíg** i ≤ n **végezd el**  **Amíg** i ≤ n **és** s[i] ≥ '0' **és** s[i] ≤ '9' **végezd el**  szám ← szám \* 10 + s[i] - '0'  i ← i + 1  **vége(amíg)**  eredmény ← eredmény + szám  szám ← 0  i ← i + 1  **vége(amíg)**  **térítsd** eredmény  **Vége(algoritmus)** |

Írjátok le a számolásKarakterekkel(s, n, i, szám) algoritmus *rekurzív* változatát úgy, hogy a fejléce és a hatása legyen azonos a fenti algoritmuséval. Az alábbi programrészletből hívjuk meg:

|  |
| --- |
| **Beolvas:** n, s  **KiÍr:** számolásKarakterekkel(s, n, 1, 0) |

1. **Iteratívból rekurzív (1)**

Legyen a következő alprogram, ahol az ***a*** és ***b*** bemeneti paraméterek természetes számok (0 < ***a*** ≤ 1 000 és 0 < ***b*** ≤ 1 000):

|  |  |
| --- | --- |
| **Algoritmus** f(a, b):  eredmény ← 0  **Amíg** a > 0 **végezd el:**  **Ha** a **mod** 2 = 1 **akkor**  eredmény ← eredmény + b  **vége(ha)**  a ← a **div** 2  b ← b + b  **vége(amíg)**  **térít** eredmény  **Vége(algoritmus)** | * 1. Adjátok meg annak a feladatnak a szövegét, amelyet ez az algoritmus old meg.   2. Mit térít az f(5, 15) hívás?   3. Írjátok le az adott algoritmus *rekurzív* változatát, amelynek fejléce azonos az iteratív (nem rekurzív) algoritmus fejlécével. |

1. **Iteratívból rekurzív (2)**

Adott a következő alprogram, ahol az ***sz*** tetszőleges természetes szám bemeneti paraméter (1 < ***sz*** ≤ 10 000):

|  |  |
| --- | --- |
| **Algoritmus** F(sz)**:**  a ← 1  b ← 0  **Amíg** sz > a + b **végezd el:**  a ← a + b  b ← a - b  **vége(amíg)**  **Ha** sz = a + b **akkor**  **térítsd true**  **különben**  **térítsd false**  **vége(ha)**  **Vége(algoritmus)** | 1. Adjátok meg annak a feladatnak a szövegét, amelyet ez az algoritmus old meg. 2. Mit térít az F(17) hívás? 3. Írjátok le az adott algoritmus *rekurzív* változatát, amelynek fejléce azonos az iteratív (nem rekurzív) algoritmus fejlécével. |

1. **Iteratívból rekurzív (3)**

Adott a következő alprogram, ahol az ***n*** és ***k*** bemeneti paraméterek (1 < ***n*** ≤ 30 000 és 1 < ***k*** ≤ 30 000) és természetes számok:

|  |  |
| --- | --- |
| **Algoritmus** F(n, k)**:**  érték ← 0  **Amíg** n ≥ k **végezd el:**  n ← n **div** k  érték ← érték + 1  **vége(amíg)**  **térít** érték  **Vége(algoritmus)** | 1. Adjátok meg annak a feladatnak a szövegét, amelyet ez az algoritmus old meg. 2. Mit térít az F(98, 2) hívás? 3. Írjátok le az adott algoritmus *rekurzív* változatát, amelynek fejléce azonos az iteratív (nem rekurzív) algoritmus fejlécével. |

1. **Előszelet (2)**

*Sors-számjegynek* hívjuk azt a természetes számot, amelyet adott természetes számra a következőképpen számítunk ki: összeadjuk a szám számjegyeit, majd a kapott összeg számjegyeit, és így tovább, amíg a kapott összeg nem válik egyszámjegyű számmá. Például, a 182 *sors-számjegye* 2 (1 + 8 + 2 = 11, 1 + 1 = 2).

Egy pontosan ***k*** számjegyű ***p*** számot egy legkevesebb ***k*** számjegyű ***q*** szám *előszeleté*nek nevezünk, ha a ***q*** szám első ***k*** számjegyéből alkotott szám (balról jobbra tekintve) egyenlő ***p***-vel. Például, 17 előszelete 174-nek, és 1713 előszelete 1 713 242-nek.

Legyen az ***sz*** természetes szám (0 < ***sz*** ≤ 30 000) és egy ***m*** soros és ***n*** oszlopos (0 < ***m*** ≤ 100, 0 < ***n*** ≤ 100) ***A*** mátrix (kétdimenziós tömb), amelynek elemei 30 000-nél kisebb természetes számok. Adva van a sorsSzámjegy(x) algoritmus, amely meghatározza az ***x*** számhoz rendelt sors-számjegyet:

|  |
| --- |
| **Algoritmus** sorsSzámjegy(x):  s ← 0  **Amíg** x **>** 0 **végezd el**  s ← s + x **MOD** 10  x ← x **DIV** 10  **Ha** x = 0 **akkor**  **Ha** s < 10 **akkor**  **térítsd** s  **különben**  x ← s  s ← 0  **vége(ha)**  **vége(ha)**  **vége(amíg)**  **térítsd s**  **Vége(algoritmus)** |

**Követelmények:**

1. Írjátok le egy *rekurzív* változatát (ismétlő struktúrák nélkül) a sorsSzámjegy(x) algoritmusnak. A fejléce és a hatása legyen azonos a fenti algoritmus fejlécével és hatásával.
2. Írjátok le a sorsSzámjegy(x) algoritmus rekurzív változatának (amelyet kidolgoztatok az **a.** pontnál) a matematikai modelljét (vagyis, írjátok le a rekurzív függvényt matematikai képlet formájában).
3. Írjatok alprogramot, amely – felhasználva a sorsSzámjegy(x) alprogramot – meghatározza az ***sz*** szám leghosszabb előszeletét(***prefix***), amelyet az adott tömb elemeinek megfelelő *sors-számjegyeiből* fel lehet építeni. Egy ilyen sors-számjegyet akárhányszor fel lehet használni. Ha nem építhető fel előszelet, ***prefix*** = -1. Az alprogram bemeneti paraméterei: ***sz***, ***m***, ***n*** és az ***A*** mátrix, kimeneti paramétere: ***prefix***.

***Példa*:** ha ***sz*** = 12319, ***m*** *=* 3, ***n*** *=* 4 és a mátrix: , akkor a leghosszabb előszelet ***prefix*** = 1231, a megfelelő sors-számjegyek pedig:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Mátrixelem értéke | 182 | 12 | 274 | 22 | 1 | 98 | 56 | 5 | 301 | 51 | 94 |
| Sors-számjegy | 2 | 3 | 4 | 4 | 1 | 8 | 2 | 5 | 4 | 6 | 4 |

1. **Bűvös számok (2)**

Legyen két természetes szám ***p*** és ***q*** (2 ≤ ***p*** ≤ 10*,*2 ≤ ***q*** ≤ 10). Egy természetes számot *bűvös*neknevezünk, ha a ***p*** számrendszerben felírt alakjában szereplő számjegyek halmaza azonos a ***q*** számrendszerben felírt alakjában szereplő számjegyek halmazával. Például. ha ***p*** = 9 és ***q*** = 7, (31)10 *bűvös szám*, mivel (34)9 = (43)7; ha ***p*** = 3 és ***q*** = 9, (9)10 *bűvös szám*, mivel (100)3 = (10)9. Adott még a számjegyek(x, b, c) alprogram, amely meghatározza az ***x*** szám számjegyeit a ***b*** számrendszerben (a ***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:**

1. 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.
2. Írjátok le a számjegyek(x, b, c) algoritmus rekurzív változatának (amelyet kidolgoztatok az **a.** pontnál) matematikai modelljét, (vagyis, írjátok le a rekurzív függvényt matematikai képlet formájában).
3. Írjatok alprogramot, amely – felhasználva a számjegyek(x, b, c) alprogramot – adott ***p*** és ***q*** számrendsze­rek ismeretében, meghatározza azt a *bűvös* számokból álló ***a*** sorozatot, amely minden 0-nál szigorúan nagyobb és adott ***n*** (1 < ***n*** ≤ 10 000) természetes számnál szigorúan kisebb számot tárol. Az alprogram bemeneti paraméterei ***p*** és ***q*** (a két alap) és az ***n*** szám. Kimeneti paraméter az ***a*** sorozat és ennek ***k*** hossza.

***Példa:*** ha ***p*** = 9, ***q*** = 7 és ***n*** = 500, az ***a*** sorozatnak ***k*** = 11 eleme lesz: (1, 2, 3, 4, 5, 6, 31, 99, 198, 248, 297).