**1.** Legyen az alábbi kifejezés, ahol ***a***természetes szám.

((a < 4) **OR** (a < 5)) **AND** (a > 2)

Az ***a*** változó mely értékeire lesz a kifejezés értéke *True*?

1. ***a*** = 3 B. ***a*** = 4 C. ***a*** = 2 D. A kifejezés értéke soha nem lesz *True*

**2.** Az alábbi algoritmus bemeneti paraméterei az ***n*** természetes szám (1 ≤ ***n*** ≤ 10000) és az ***n*** elemű, természetes számokat tároló ***v*** sorozat (***v***[1], ***v***[2], …, ***v***[***n***]).

**Algorithm** f(v, n):

    x ← 0

**For** i ← 1, n **execute**

        c ← v[i]

**While** c **MOD** 3 = 0 **execute**

            x ← x + 1

            c ← c **DIV** 3

**EndWhile**

**EndFor**

**return** x

**EndAlgorithm**

Állapítsátok meg, hogy az alábbi állítások közül melyek igazak?

* 1. Az algoritmus visszatéríti a ***v*** sorozat 3-mal osztható elemeinek darabszámát
  2. Az algoritmus visszatéríti azt a legnagyobb ***k*** számot, amelyre ***v***[1]\****v***[2]\*…\****v***[***n***] osztható 3***k***-nal
  3. Az algoritmus visszatéríti azt a legnagyobb ***k*** számot, amelyre ***v***[1]+***v***[2]+…+***v***[***n***] osztható 3***k***-nal
  4. Az algoritmus visszatéríti a ***v*** sorozat 3-mal osztható elemeinek összegét

**3.** Legyen az alábbi kifejezés, ahol ***x*** egy nullától különböző természetes szám.

(x **MOD** 2) + ((x + 1) **MOD** 2)

Állapítsátok meg, hogy az alábbi állítások közül melyek igazak?

1. A kifejezés értéke 1 bármely ***x*** nullától különböző természetes szám esetében.
2. A kifejezés értéke 1 akkor és csakis akkor, ha ***x*** páros szám.
3. A kifejezés értéke 1 akkor és csakis akkor, ha ***x*** páratlan szám.
4. Létezik olyan ***x*** természetes szám, amelyre a kifejezés értéke szigorúan nagyobb, mint 1.

**4.** Legyen a feldolgoz(x, n) algoritmus, amelynek bemeneti paraméterei az ***n*** természetes szám (1 ≤ ***n*** ≤ 10000) és az ***n*** elemű, természetes számokat tároló ***x*** sorozat (***x***[1], ***x***[2], …, ***x***[***n***]). A **/** operátor a valós osztást jelöli (ex. 3 / 2 = 1,5).

|  |  |
| --- | --- |
| **Algorithm** feldolgoz(x, n):      p ← 1  **For** k ← 1, n - 1 **execute**  p← p + 1  **For** i ← 1, n - 1 **execute**  **If** x[i] > x[i + 1] **then**                  x[i] ← x[i] \* x[i + 1]                  x[i + 1] ← x[i] **/** x[i + 1]                  x[i] ← x[i] **/** x[i + 1]  **EndIf**  **EndFor**  **EndFor**  n ← p  **EndAlgorithm** | Az alábbi állítások közül, melyik írja le az ***x*** sorozat módosulását? (Az algoritmust feldolgoz(x, n) alakban hívjuk meg.)   1. Az ***x*** sorozat elemei nem változnak meg 2. Az ***x*** sorozat elemei csökkenően rendezettekké válnak 3. Az ***x*** sorozat elemei növekvően rendezettekké válnak 4. Az ***n*** szám értéke csökken 1-gyel |

**5.** Legyen a számol(a, n) algoritmus, amelynek bemeneti paraméterei az ***n*** természetes szám (1 ≤ ***n*** ≤ 10000) és az ***n*** elemű, természetes számokat tároló ***a*** sorozat (***a***[1], ***a***[2], …, ***a***[***n***]).

**Algorithm** számol(a, n):

**If** n = 0 **then**

**return** 0

**else**

**return** a[n] \* (a[n] **MOD** 2) + számol(a, n - 1)

**EndIf**

**EndAlgorithm**

***n*** és ***a*** mely értékeire térít vissza a számol(a, n) algoritmus 10-et?

A. ***n*** = 4, ***a*** = (2, 4, 7, 5) B. ***n*** = 6, ***a*** = (3, 1, 2, 5, 8, 1)

1. ***n*** = 6, ***a*** = (2, 4, 5, 3, 8, 5) D. ***n*** = 7, ***a*** = (1, 1, 2, 1, 1, 1, 3)

**6.** Legyen a számol(v, n) algoritmus, amelynek bemeneti paraméterei az ***n*** természetes szám (1 ≤ ***n*** ≤ 10000) és az ***n*** elemű, természetes számokat tároló ***v*** sorozat (***v***[1], ***v***[2], …, ***v***[***n***].

**Algorithm** számol(v, n):

    m ← 0; x ← 0; s ← 0

**For** i ← 1, n **execute**

        s ← s + v[i]

        m ← m + (s **MOD** 2 + x) **MOD** 2

        x ← s **MOD** 2

**EndFor**

**return** m

**EndAlgorithm**

Állapítsátok meg, hogy az alábbi állítások közül melyek igazak?

1. Az algoritmus kiszámítja és visszatéríti a ***v*** sorozat páratlan szám értékű elemeinek összegét
2. Az algoritmus kiszámítja és visszatéríti a ***v*** sorozat páros szám értékű elemeinek összegét
3. Az algoritmus kiszámítja és visszatéríti a ***v*** sorozat páratlan szám értékű elemeinek darabszámát
4. Az algoritmus kiszámítja és visszatéríti a ***v*** sorozat páros szám értékű elemeinek darabszámát

**7.** Legyen a bűvös(x) algoritmus, ahol ***x*** természetes szám (1 ≤ ***x*** ≤ 32000).

**Algorithm** bűvös(x):

    bal ← 1; jobb ← x

**While** bal ≤ jobb **execute**

        közép ← (bal + jobb) **DIV** 2

**If** közép \* közép = x **then**

**return** ***True***

**EndIf**

**If** közép \* közép < x **then**

            bal ← közép + 1

**else**

            jobb ← közép – 1

**EndIf**

**EndWhile**

**return** ***False***

**EndAlgorithm**

Állapítsátok meg, hogy az alábbi állítások közül melyek igazak?

1. Az algoritmus vizsgálja, hogy létezik-e ***x***-nél kisebb négyzetszám
2. Az algoritmus megszámolja az ***x*** szám prímosztóit
3. Az algoritmus vizsgálja, hogy az ***x*** szám prímszám-e
4. Az algoritmus vizsgálja, hogy az ***x*** szám négyzetszám-e

**8.** Legyen a mitCsinál(n) algoritmus, ahol ***n*** este természetes szám (1 ≤ ***n*** ≤ 10000).

**Algorithm** mitCsinál(n):

    a ← n

    b ← 0

**While** a ≠ 0 **execute**

      b ← b \* 10 + a **MOD** 10

      a ← a **DIV** 10

**EndWhile**

**If** n = b **then**

**return** ***True***

**else**

**return *False***

**EndIf**

**EndAlgorithm**

Állapítsátok meg, hogy az alábbi állítások közül melyek igazak?

1. Az algoritmus vizsgálja, hogy az ***n*** prímszám-e
2. Az algoritmus vizsgálja, hogy az ***n*** palindromszám-e
3. Az algoritmus mindig *True*-t térít vissza
4. Az algoritmus vizsgálja, hogy az ***n*** osztható-e 10-zel

**9.** Az alábbi algoritmusokban ***x*** egy ***n*** elemű, nem feltétlenül különböző egész számokat tároló vektor (***x***[1], ***x***[2], …, ***x***[***n***]). Mely algoritmus alakítja halmazzá a vektort?

|  |  |
| --- | --- |
| A.  **Algorithm** alakít(x, n):  i ← 1  **While** i < n **execute**  j ← i + 1  **While** j ≤ n **AND** x[i] ≠ x[j] **execute**  j ← j + 1  **EndWhile**  **If** j ≤ n **then**  x[j] ← x[n-1]  n ← n - 1  i ← i + 1  **EndIf**  **EndWhile**  **EndAlgorithm** | B.  **Algorithm** alakít(x, n):  i ← 1  **While** i < n **execute**  j ← i + 1  **While** j ≤ n **AND** x[i] ≠ x[j] **execute**  j ← j + 1  **EndWhile**  **If** j ≤ n **then**  x[j] ← x[n-1]  n ← n - 1  **EndIf**  i ← i + 1  **EndWhile**  **EndAlgorithm** |
| C.  **Algorithm** alakít(x, n):  i ← 1  **While** i < n **execute**  j ← i + 1  **While** j ≤ n **AND** x[i] ≠ x[j] **execute**  j ← j + 1  **EndWhile**  **If** j ≤ n **then**  x[j] ← x[n-1]  **else**  i ← i + 1  n ← n - 1  **EndIf**  **EndWhile**  **EndAlgorithm** | D.  **Algorithm** alakít(x, n):  i ← 1  **While** i < n **execute**  j ← i + 1  **While** j ≤ n **AND** x[i] ≠ x[j] **execute**  j ← j + 1  **EndWhile**  **If** j ≤ n **then**  x[j] ← x[n]  n ← n - 1  **else**  i ← i + 1  **EndIf**  **EndAlgorithm** |

**10.** Legyen minden olyan sorozat, amelynek a hossza rendre ***h*** ϵ {1, 2, 3} és elemei az {*a*, *b*, *c*, *d*, *e*} halmazból valók. Hány olyan sorozat van ezek között, amelyek növekvően rendezettek és páratlan számú magánhangzót tartalmaznak?

A. 7 B. 20 C. 14 D. 78

**11.** Legyen a számol(a, b) algoritmus, ahol ***a*** és ***b*** természetes számok (1 ≤ ***a***, ***b*** ≤ 10000).

**Algorithm** számol(a, b):

    x ← 1

**For** i ← 1, b **execute**

        x ← (x **MOD** 10) \* a

**EndFor**

**return** x

**EndAlgorithm**

Állapítsátok meg, hogy az alábbi állítások közül melyek igazak?

1. Ha ***a*** = 2021 és ***b*** = 2021, az algoritmus 2021-et térít vissza
2. Ha ***a*** = 2021 és 1 ≤ ***b*** ≤ 10000, az algoritmus minden esetben 2021-et térít vissza
3. Ha ***a*** = 7777 és ***b*** = 2021, az algoritmus 7777-et térít vissza
4. Ha 1 ≤ ***a*** ≤ 10000 és ***b*** = 2021, az algoritmus minden esetben ***a*** értékét téríti vissza

**12.** Hány elem található összesen egy ***n*** soros és ***n*** oszlopos négyzetes mátrix két átlóján (10 ≤ ***n*** ≤ 1000)?

1. 2 \* ***n***
2. ***n*** \* ***n***
3. 2 \* ***n*** – 1
4. 2 \* ***n*** – (***n*** **MOD** 2)

**13.** A következő kifejezések körül melyeknek az értéke *True,*amikor ***a*** = 1 és ***b*** = 0?

1. **NOT** (((a > 0) **AND** (b < 1)) **OR** (a > 1))
2. ((b > 0) **AND** (b < 1)) **OR** ((a > 0) **AND** (a < 2))
3. (**NOT** (a > b)) **OR** (**NOT** (b > 0))
4. (a > 0) **OR** ((b > 0) **AND** (b < 0)) **OR** (a < 1)

**14.** A számol[i](e, n) algoritmusok 1 ≤ ***i*** ≤ 4, bemeneti paraméterei az *n* soros és ***n*** oszlopos ***e*** mátrix (***e***[1][1], …, ***e***[1][***n***], ***e***[2][1], …, ***e***[***n***][***n***]) és az ***n***  természetes szám (1 ≤ ***n*** ≤ 1000). Válasszátok ki azt a számol[i] algoritmust, amelynek eredménye különbözik a másik három algoritmus eredményétől, vagyis számol[i](e, n) ≠ számol[j](e, n) ***e***, ***n***, ***j***, 1 ≤ ***j*** ≤ 4, ***i*** ≠ ***j***.

|  |  |
| --- | --- |
| **Algorithm** számol[1](e, n):      s ← 0  **For** i ← 1, n **execute**  s ← s + e[i][i]  **EndFor**  **return** s  **EndAlgorithm** | **Algorithm** számol[2](e, n):      s ← 0  **For** i ← 1, n **execute**  **For** j ← 1, n, **execute**  **If** i = j **then**  s ← s + e[i][j]  **EndIf**  **EndFor**  **EndFor**  **return** s  **EndAlgorithm** |

|  |  |
| --- | --- |
| **Algorithm** számol[3](e, n):      s ← 0      i ← 1  **While** i ≤ n **execute**  s ← s + e[i][i]          i ← i + 1  **EndWhile**  **return** s  **EndAlgorithm** | **Algorithm** számol[4](e, n):      s ← 0  **For** i ← 1, n **execute**  **For** j ← i + 1, n, **execute**  **If** i = j **then**  s ← s + e[i][j]  **EndIf**  **EndFor**  **EndFor**  **return** s  **EndAlgorithm** |

**15.** Legyen a mitCsinál(a, b) algoritmus, ahol ***a*** és ***b*** természetes számok (1 ≤ ***a*** < ***b*** ≤ 10000). 

**Algorithm** mitCsinál(a, b):

    m ← a

**While** b **MOD** m > 0 **execute**

        m ← m + 1

**EndWhile**

**return** m

**EndAlgorithm**

Mit fog visszatéríteni a mitCsinál(47, 100) hívás?

1. 48 B. 50 C. 3 D. 100

**16.** Legyen az kiÍr(n) algoritmus, ahol ***n*** természetes szám (0 ≤ ***n*** ≤ 10000).

**Algorithm** kiÍr(n):

**Write** n

**If** n > 0 **then**

        kiÍr(n - 1)

**Write** n

**EndIf**

**EndAlgorithm**

Mit fog kiírni, ha kiÍr(4) alakban hívjuk meg?

1. 432100123 B. 123401234 C. 1234004321 D. 432101234

**17.** A következő ***x*** számrendszeralapok közülmelyikre érvényes a 232(***x***) ≤ 67(10) reláció?

1. ***x*** = 5 B. ***x*** = 3 C. ***x*** = 4 D. ***x*** = 6

**18.** Legyen az f(a, b) algoritmus, ahol ***a*** és ***b*** egész számok (-10000 ≤ ***a***, ***b*** ≤ 10000).

**Algorithm** f(a, b):

**Scrie** “BRAVO”

**If** (a = 0) **OR (**b = 0) **then**

**return** 1

**EndIf**

**If** a > b **then**

**return** f(a - b \* b, a \* (a - b) - b \* (a - b))

**EndIf**

**If** a ≤ b **then**

**return** f(b - a \* a, a \* (a - b) - b \* (a - b))

**EndIf**

**EndAlgorithm**

Hányszor írja ki az algoritmus a BRAVO szöveget az f(f(3, 2), f(2, 3)) hívás eredményeként?

1. 8-szor B. 6-szor C. 3-szor D. végtelenszer

**19.** Legyen az ***X*** = 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7… sorozat, amelyben minden ***n*** szám ***n***-szer fordul elő egymásutáni pozíciókon. Tudva, hogy az első elem az első pozíción található, mely pozíciókon fordul elő a 21?

1. A [210, 230] intervallum pozícióin
2. A [211, 231] intervallum pozícióin
3. A [212, 232] intervallum pozícióin
4. A [209, 229] intervallum pozícióin

**20.** Legyen a mitCsinál(n, i) algoritmus, ahol ***n*** természetes szám (2 ≤ ***n*** ≤ 1000).

**Algorithm** mitCsinál(n, i):

**If** i \* i > n **then**

**return** 0

**EndIf**

**If** i \* i = n **then**

**return** i

**EndIf**

**If** n **MOD** i = 0 **then**

**return** i + n **DIV** i + mitCsinál(n, i + 1)

**else**

**return** mitCsinál(n, i + 1)

**EndIf**

**EndAlgorithm**

Állapítsátok meg, hogy az alábbi állítások közül melyek igazak, ha mitCsinál(n, 2) alakban hívjuk meg?

1. Az algoritmus kiszámítja és visszatéríti az ***n*** szám valódi osztói összegének kétszeresét.
2. Az algoritmus kiszámítja és visszatéríti az ***n*** szám valódi osztóinak összegét.
3. Az algoritmus kiszámítja és visszatéríti az ***n*** szám valódi és nem valódi osztóinak összegét.
4. Az algoritmus vizsgálja, hogy ***n*** négyzetszám-e. Ha igen, visszatéríti az ***n*** szám négyzetgyökét, különben 0-át térít vissza.

**21.** A költöztet(a, n) algoritmus paraméterei az ***n*** elemű, egész számokat tároló ***a*** sorozat, (***a***[1], ***a***[2], …, ***a***[***n***]) és az ***n*** (1 ≤ ***n*** ≤ 10000) természetes szám. Az algoritmus elköltözteti a 0 értékű elemeket a sorozat végére, de megőrzi a nullától különböző elemek sorrendjét. Például, ha ***a*** = (4, 0, 2, 5, 1, 0, 7, 11, 0, 3), az algoritmus végrehajtása után az ***a*** sorozat elemei (4, 2, 5, 1, 7, 11, 3, 0, 0, 0) lesznek. Melyik helyes a következő algoritmusok közül?

|  |  |
| --- | --- |
| **Algorithm** költöztet(a, n):      s ← ***TRUE***  **While** s = ***TRUE*** **execute**  s ← ***FALSE***  **For** i ← 1, n - 1 **execute**   **If** a[i] = 0 **then**                 tmp ← a[i]                 a[i] ← a[i + 1]                 a[i + 1] ← tmp                 s ← ***TRUE***  **EndIf**  **EndFor**  **EndWhile**  **EndAlgorithm** | **Algorithm** költöztet(a, n):      c ← 0  **For** i ← 0, n **execute**  **If** a[i] = 0 **then**              c ← c + 1  **EndIf**  **EndFor**      i ← n  **While** c > 0 **execute**          a[i] ← 0          i ← i - 1          c ← c - 1  **EndWhile**  **EndAlgorithm** |

|  |  |
| --- | --- |
| **Algorithm** költöztet(a, n):      d ← 0      i ← 1  **While** i + d ≤ n **execute**  **While** i+d ≤ n **AND** a[i+d] = 0 **execute**              d ← d + 1  **EndWhile**  **If** i + d ≤ n **then**            a[i] ← a[i + d]            i ← i + 1  **EndIf**  **EndWhile**  **While** i ≤ n **execute**          a[i] ← 0          i ← i + 1  **EndWhile**  **EndAlgorithm** | **Algorithm** költöztet(a, n):      i ← 1      f ← n  **While** i < f **execute**  **While** i< f **AND** a[i] ≠ 0 **execute**              i ← i + 1  **EndWhile**  **While** i < f **AND** a[f] = 0 **execute**              f ← f - 1  **EndWhile**  **If** i < f **then**              tmp ← a[i]              a[i] ← a[f]              a[f] ← tmp  **EndIf**  **EndWhile**  **EndAlgorithm** |

**22.** Legyen a következő, természetes számokat tároló sorozat: 1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, ....

Melyik szám lesz a sorozat 10-edik eleme?

A. 141122221

B. 13211311123113112211

C. 411211131221

D. 1114122112132113212221

**23.** Legyen a mitCsinál(T, n, e) algoritmus, ahol ***T*** egy ***n*** elemű, növekvően rendezett, természetes számokat tároló sorozat (***T***[1], ***T***[2], …, ***T***[***n***]) és ***n*** és ***e*** (1 ≤ ***n***, ***e*** ≤ 10000) természetes számok.

**Algorithm** mitCsinál(T, n, e):

**If** e **MOD** 2 = 0 **then**

        a ← 1;  b ← n

**While** a ≤ b **execute**

            m ← (a + b) **DIV** 2

**If** e < T[m] **then**

                b ← m - 1

**else**

**If** e > T[m] **then**

                    a ← m + 1

**else**

**return** ***True***

**EndIf**

**EndIf**

**EndWhile**

**return** ***False***

**else**

        c ← 1

**While** c ≤ n **execute**

**If** e = T[c] **then**

**return*****True***

**EndIf**

            c ← c + 1

**EndWhile**

**return** ***False***

**EndIf**

**EndAlgorithm**

Állapítsátok meg, hogy az alábbi állítások közül melyek igazak?

1. Az algoritmus **nem** vizsgálja, hogy ***e*** a ***T*** sorozat páros indexű pozícióján található.
2. Az algoritmus vizsgálja, hogy a ***T*** sorozatban megtalálható-e az ***e*** szám és ha ***e*** páratlan szám, a keresőalgoritmus, amit alkalmaz, a bináris keresés algoritmusa.
3. Az algoritmus vizsgálja, hogy a ***T*** sorozatban megtalálható-e az ***e*** szám és ha ***e*** páros szám, a keresőalgoritmus, amit alkalmaz, a bináris keresés algoritmusa.
4. Az algoritmus csak akkor vizsgálja, hogy a ***T*** sorozatban megtalálható-e az ***e***, ha ***e*** páratlan szám.

**24.** Egyenlő szárú háromszögeket szeretnénk rajzolni csillag (\*) és pont (.) karakterek segítségével.Az alábbi példa egy ilyen háromszög, ha ***n*** = 5. A rajzban felhasználtunk 12 csillagot és 23 pontot.

....\*

...\*.\*

..\*...\*

.\*.....\*

\*.\*.\*.\*.\*

Állapítsátok meg, hogy az alábbi állítások közül melyek igazak?

1. Ha ***n*** = 2, a rajzhoz pontosan 3 csillag és 4 pont szükséges.
2. Ha ***n*** = 7, a rajzhoz pontosan 18 csillag és 52 pont szükséges.
3. Ha ***n*** = 7, a rajzhoz pontosan 18 csillag és 48 pont szükséges.
4. Ha ***n*** = 15, a rajzhoz pontosan 42 csillag és 288 pont szükséges.

**25.** Legyen a összeg(n, t, maxÖsszeg, eleje, vége) algoritmus, ahol ***n*** este természetes szám (2 ≤ ***n*** ≤ 10000) és ***t*** egy ***n*** elemű egész számokat tároló sorozat (***t***[1], ***t***[2], …, ***t***[***n***]). Az algoritmus célja meghatározni azt a leghosszabb tömbszakaszt, amelynek összege maximális. A ***maxÖsszeg***, ***eleje*** és ***vége*** kimeneti paraméterek (beazonosítják a keresett tömbszakaszt).

**Algorithm** összeg(n, t, maxÖsszeg, eleje, vége):

maxÖsszeg ← t[1]

sum ← maxÖsszeg

eleje ← 1

vége ← 1

start ← 1

**For** i = 2, n **execute**

**If** sum < 0 **then**

sum ← t[i]

start ← i

**else**

sum ← sum + t[i]

**If** maxÖsszeg < sum **then**

maxÖsszeg ← sum

eleje ← start

vége ← i

**else**

**If** maxÖsszeg = sum **AND** i - start > vége – eleje **then**

eleje ← start

vége ← i

**EndIf**

**EndIf**

**EndIf**

**EndFor**

**EndAlgorithm**

Az algoritmus akkor és csakis akkor működik helyesen, ha:

A. A negatív számok előtt és után található legalább egy pozitív szám.

B. A pozitív elemek összege nagyobb, mint a negatív elemek összege.

C. Bármely negatív szám abszolútértéke kisebb, mint az előtte található pozitív számok összege.

D. A sorozatban létezik legalább egy pozitív szám.

**26.** Legyen az átRendez(n, a) algoritmus, ahol ***n*** természetes szám (1 ≤ ***n*** ≤ 10000) és ***a*** egy **2*n*** elemű, természetes számokat tároló sorozat (***a***[1], ***a***[2], …, ***a***[**2*n***]). Tudjuk még, hogy a sorozat páros szám értékű elemeinek darabszáma egyenlő a páratlan szám értékű elemek darabszámával.

A következő algoritmusok közül melyik rendezi át a sorozat elemeit úgy, hogy a páratlan szám értékű elemeknek az indexei páratlan számok legyenek és a páros szám értékű elemeknek az indexei páros számok legyenek?



**Algorithm** átRendez(n, a):

**For** i ← 1, 2 \* n - 1 **execute**

**If** a[i] **MOD** 2 ≠ i **MOD** 2 **then**

**For** j ← i + 1, 2 \* n **execute**

**If** a[j] **MOD** 2 ≠ j **MOD** 2 **then**

a[i] ← a[i] + a[j];a[j] ← a[i] - a[j]; a[i] ← a[i] - a[j]

**EndIf**

**EndFor**

**EndIf**

**EndFor**

**EndAlgorithm**



**Algorithm** átRendez(n, a):

**For** i ← 1, 2 \* n - 1 **execute**

**If** a[i] **MOD** 2 ≠ i **MOD** 2 **then**

**For** j ← i + 1, 2 \* n **execute**

**If** (a[i] **MOD** 2 ≠ i **MOD** 2) **AND** (a[j] **MOD** 2 ≠ j **MOD** 2) **then**

a[i] ← a[i] + a[j];a[j] ← a[i] - a[j];a[i] ← a[j] - a[i]

**EndIf**

**EndFor**

**EndIf**

**EndFor**

**EndAlgorithm**



**Algorithm** átRendez(n, a):

**For** i ← 1, 2 \* n - 1 **execute**

**If** a[i] **MOD** 2 ≠ i **MOD** 2 **then**

**For** j ← i + 1, 2 \* n **execute**

**If** (a[i] **MOD** 2 ≠ i **MOD** 2) **AND** (a[j] **MOD** 2 ≠ j **MOD** 2) **AND**

(a[i] **MOD** 2 ≠ a[j] **MOD** 2) **then**

a[i] ← a[i] + a[j]; a[j] ← a[i] - a[j]; a[i] ← a[i] - a[j]

**EndIf**

**EndFor**

**EndIf**

**EndFor**

**EndAlgorithm**



**Algorithm** átRendez(n, a):

**For** i ← 1, 2 \* n - 1 **execute**

**For** j ← i + 1, 2 \* n **execute**

**If** (a[j] **MOD** 2 = 0) **AND** (a[j] **MOD** 2 ≠ 0) **OR** (a[j] **MOD** 2 ≠ 0) **AND**

            (a[j] **MOD** 2 = 0) **then**

a[i] ← a[i] + a[j];a[j] ← a[i] - a[j];a[i] ← a[i] - a[j]

**EndIf**

**EndFor**

**EndFor**

**EndAlgorithm**

**27.** Az egyszerűsít(nr, num) algoritmus meghatározza az ***aux*1 / *aux*2** nem egyszerűsíthető törtet, ahol ***aux*1 / *aux*2 = *nr* / *num*** (***aux*1**, ***aux*2**, ***nr***, ***num*** természetes számok és ***num*, *aux*2** ≠ 0).

**Algorithm** egyszerűsít(nr, num):

    d ← függvény(nr, num)

    aux1 ← nr **DIV** d

    aux2 ← num **DIV** d

**EndAlgorithm**

Az alábbi függvény(a, b) algoritmusok közül melyek helyesek?

|  |  |
| --- | --- |
| **Algorithm** függvény(a, b):      d ← 1  **While *True*****execute**  **If** a **MOD** d = 0 **AND** b **MOD** d = 0 **then**  **return** d  **EndIf**          d ← d + 1  **EndWhile**  **EndAlgorithm** | **Algorithm** függvény(a, b):  **While** b ≠ 0 **execute**          c ← a **MOD** b          a ← b          b ← c  **EndWhile**    **return** a  **EndAlgorithm** |
| **Algorithm** függvény(a, b):  **While** a ≠ b **execute**  **If** a > b **then**              a ← a - b  **else**              b ← b - a  **EndIf**  **EndWhile**    **return** a  **EndAlgorithm** | **Algorithm** függvény(a, b):      d ← a  **While (**a **MOD** d ≠ 0) **OR** (b **MOD** d ≠ 0)          d ← d - 1  **EndWhile**    **return** d  **EndAlgorithm** |

**28.** Szeretnénk felosztani ***k***, egymást érintő, egyenlő hosszúságú tömbszakaszra egy ***n*** elemű sorozatot(1 ≤ ***n*** ≤ 1000, 1 ≤ ***k*** ≤ ***n***) úgy, hogy a sorozat minden eleme pontosan egy tömbszakaszhoz tartozzon. Ha ***n*** nem osztható ***k***-val, elfogadjuk azt a felosztást, amelynek megfelelően bármely két tömbszakasz hosszának különbsége legtöbb 1. Megadjuk négyféle módját annak, ahogy minden ***j*** (1 ≤ ***j*** ≤ ***k***) tömbszakasz első elemének indexe kiszámítható. Melyik a helyes?

1. ((j \* n - 1) **DIV** k) - 1
2. ((j - 1) \* n ) **DIV** k + 1
3. (j - 1) \* (n **DIV** k)
4. ((j - 1) \* n + k) **DIV** k

**29.** Adott egy ***n*** (pontosan kilenc, különböző értékű számjegyből álló) természetes szám és a következő algoritmusok:

* felCserél(a, b) – felcseréli az ***a*** és ***b*** természetes számokat;
* épít(n, x, érték) – az ***érték*** értékű számhoz hozzáragasztja az ***n*** számjegyet tároló ***x*** sorozatban található számjegyeket (például, ha ***n*** = 4, ***x*** = (2, 6, 0, 4) és ***érték*** = 71, a épít(n, x, érték) algoritmus végrehajtása után ***érték*** = 712604 lesz).

Állapítsátok meg, hogy az alábbi algoritmusok közül melyik határozza meg azt az ***m*** számot, amelyik a legkisebb az ***n***-nél nagyobb számok között és *pontosan az* ***n*** számjegyeit tartalmazza. Ha nincs ilyen érték, az eredmény -1.

|  |  |
| --- | --- |
| A.  **Algorithm** meghatároz(n)  poz ← 1  v[poz] ← n **MOD** 10  n ← n **DIV** 10  kész ← ***False***  **While** **NOT** kész **AND** n > 0 **execute**  poz ← poz + 1  v[poz] ← n **MOD** 10  n ← n **DIV** 10  **If** v[poz] < v[poz - 1] **then**  kész ← ***True***  **EndIf**  **EndWhile**  **If** n = 0 **then**  **return** -1  **EndIf**  felCserél(v[poz], v[poz - 1])  n ← n \* 10 + v[poz]  épít(poz - 1, v, n)  **return** n  **EndAlgorithm** | B.  **Algorithm** meghatároz(n)  poz ← 0  **While** n > 0 **execute**  poz ← poz + 1  v[poz] ← n **MOD** 10  n ← n **DIV** 10  **EndWhile**  kész ← ***False***  i ← poz  **While NOT** kész **AND** i > 0 **execute**  **If** v[i] < v[i - 1] **then**  kész ← ***True***  **EndIf**  i ← i - 1  **EndWhile**  felCserél(v[i], v[i - 1])  n ← n \* 10 + v[poz]  épít(poz, v, n)  **return** n  **EndAlgorithm** |

C.

**Algorithm** meghatároz(n)

poz ← 0

**While** n **DIV** 10 ≠ 0 **AND** n **MOD** 10 < n **DIV** 10 **MOD** 10 **execute**

poz ← poz + 1

v[poz] ← n **MOD** 10

n ← n **DIV** 10

**EndWhile**

**If** n < 10 **then**

**return** -1

**else**

poz ← poz + 1

v[poz] ← n **MOD** 10

n ← n **DIV** 10

c ← n **MOD** 10

n ← n **DIV** 10

felCserél(v[poz], c)

n ← n \* 10 + c

épít(poz, v, n)

**return** n

**EndIf**

**EndAlgorithm**

D.

**Algorithm** meghatároz(n)

poz ← 0

másolat ← n

**While** n > 0 **execute**

poz ← poz + poz

v[poz] ← n **MOD** 10

n ← n **DIV** 10

**EndWhile**

kész ← ***False***

i ← poz

**While NOT** kész **AND** i > 1 **execute**

**If** v[i] > v[i - 1] **then**

kész ← ***True***

**else**

i ← i - 1

**EndIf**

**EndWhile**

**If** i = 0 **then**

**return** -1

**EndIf**

felCserél(v[i - 2], v[i - 1])

p ← 1

**For** k = 1, i – 1 **execute**

p ← p \* 10

**EndFor**

n ← másolat / p

épít(i - 1, v, n)

**return** n

**EndAlgorithm**

**30.** Legyen a előszelet(n) algoritmus, amely – adott ***n*** (9 < ***n*** < 1020 – 1) természetes szám esetében – megkeresi az ***n*** számnak azt a leghosszabb előszeletét, amely megtalálható a szám belsejében is (kivéve az első és utolsó számjegyet). Az algoritmus visszatéríti ennek az előszeletnek a hosszát. Például, ha ***n*** = 12133121, az előszelet 12 és a hossza 2; ha ***n*** = 34534536, az előszelet 3453 és a hossza 4; ha ***n*** = 1223, nincs kért tulajdonságú előszelet (0 hosszúnak tekintjük).

A következő algoritmusok közül melyik helyes?

|  |  |
| --- | --- |
| **Algorithm** előszelet(n):  nr ← n; c ← 0;p ← 1  **While** nr > 0 **execute**  c ← c + 1  nr ← nr **DIV** 10  p ← p \* 10  **EndWhile**  f1 ← 100;f2 ← p **DIV** 100;k ← 1  ok ← 0  **While** ok = 0 **execute**  n1 ← n **DIV** f1; f3 ← 10  **For** i ← 1, k **execute**  n2 ← (n **DIV** f3) **MOD** f2  **If** n1 = n2 **then**  ok ← 1  **return** c - k - 1  **EndIf**  f3 ← f3 \* 10  **EndFor**  f1 ← f1 \* 10  f2 ← f2 **DIV** 10  k ← k + 1  **EndWhile**  **return** -1  **EndAlgorithm** | **Algorithm** előszelet**(**n**):**      // a c vektor inicializálása    nr ← n      p ← 0    **While** nr > 0 **execute**          c[p + 1] ← nr **MOD** 10          nr ← nr **DIV** 10      p ← p + 1    **EndWhile**    **For** i ← 1, p - 2      **For** j ← p – 1, i + 1, -1 **execute**        ok ← 1        **For** k ← 0, i - 1 **execute**          **If** c[p - 1 - k] ≠ c[j - k] **then**          ok ← 0          **EndIf**        **EndFor**        **If** ok = 1 **then**          **return** i        **EndIf**      **EndFor**    **EndFor**    **return** -1  **EndAlgorithm** |
| **Algorithm** előszelet(n):      // a c vektor inicializálása      nr ← n      p ← 0  **While** nr > 0 **execute**          c[p + 1] ← nr **MOD** 10          nr ← nr **DIV** 10          p ← p + 1  **EndWhile**  **For** i ← p - 2, 1, -1 **execute**  **For** j ← p – 1, i + 1, -1 **execute**              ok ← 1  **For** k ← 0, i - 1 **execute**  **If** c[p-k] ≠ c[j-k] **then**                      ok ← 0  **EndIf**  **EndFor**  **If** ok = 1 **then**  **return** i  **EndIf**  **EndFor**  **EndFor**  **return** 0  **EndAlgorithm** | **Algorithm** előszelet(n):      nr ← n; c ← 0; p ← 1  **While** nr > 0 **execute**          c ← c + 1          nr ← nr **DIV** 10          p ← p \* 10  **EndWhile**      f1 ← p **DIV** 10;  f2 ← 10      k  ← c – 2;  ok ← -1  **For** t ← 1, c - 2 **execute**          n1 ← n **DIV** f1; f3 ← 10  **For** i ← 1, k **execute**              n2 ← (n **DIV** f3) **MOD** f2  **If** n1 = n2 **then**  **If** ok < c - k - 1 **then**                      ok ←  c - k - 1  **EndIf**  **EndIf**              f3 ← f3 \* 10  **EndFor**          f1 ← f1 **DIV** 10          f2 ← f2 \* 10          k ← k - 1  **EndFor**    **If** ok < 0 **then:**      **return** 0  **EndIf**  **return** ok  **EndAlgorithm** |