**1.** Adott az f(x, b) algoritmus , ahol ***x*** és ***b*** természetes számok (1 ≤ ***x*** ≤ 1000, 1 < ***b*** ≤ 10).

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

s ← 0

**While** x > 0 **execute**

s ← s + x **MOD** b

x ← x **DIV** b

**EndWhile**

**return** s **MOD** (b - 1) = 0

**EndAlgorithm**

Adjátok meg az algoritmus hatását:

vizsgálja, hogy az ***x*** szám számjegyeinek összege osztható-e ***b*** – 1-gyel

vizsgálja, hogy az ***x*** szám ***b*** – 1 számrendszerbeli ábrázolásában a számjegyek összege osztható-e ***b*** – 1-gyel

vizsgálja, hogy az ***x*** természetes szám osztható-e ***b*** – 1-gyel

vizsgálja, hogy az ***x*** szám ***b*** számrendszerbeli ábrázolásában a számjegyek összege osztható-e ***b*** – 1-gyel

**2.** Adott a calcul(a, b) algoritmus, ahol ***a*** és ***b*** természetes számok, 1 ≤ ***a*** ≤ 1 000, 1 ≤ ***b*** ≤ 1 000.

1. **Algorithm** calcul(a, b):
2. **If** a ≠ 0 **then**
3. **return** calcul(a **DIV** 2, b + b) + b \* (a **MOD** 2)
4. **EndIf**
5. **return** 0
6. **EndAlgorithm**

Az alábbi állítások közül, melyek **NEM** helyesek?

ha ***a*** egyenlő ***b***-vel, az algoritmus ***a*** értékét téríti vissza

ha ***a*** = 1000 és ***b*** = 2, az algoritmus 10-szer hívja meg önmagát

az algoritmus az ***a*** / 2 + 2 \* ***b*** kifejezés értékét számítja ki és téríti vissza

az 5. sorban található utasítás egyszer hajtódik végre

**3.** Adott az (1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 2, 5, 11, ...) sorozat, amelyet a következőképpen építettünk fel: kiindulunk a természetes számok sorozatából, és a nem prímszámokat behelyettesítjük az illető szám valódi osztóival, mindegyik ***d*** osztót csak egyszer írva a sorozatba egy-egy szám esetében. A következő algoritmusok közül melyik határozza meg ennek a sorozatnak az ***n***-dik elemét (***n*** természetes szám, 1 ≤ ***n*** ≤ 1000)?

|  |  |
| --- | --- |
| A.  **Algorithm** identificare(n):  a **←** 1; b **←** 1; c **←** 1  **While** c < n **execute**  a **←** a + 1  b **←** a  c **←** c + 1  d **←** 2  **While** c ≤ n **AND** d ≤ a **DIV** 2 **execute**  **If** a **MOD** d **=** 0 **then**  c **←** c + 1  b **←** d  **EndIf**  d **←** d + 1  **EndWhile**  **EndWhile**  **return** b  **EndAlgorithm** | B.  **Algorithm** identificare(n):  a **←** 1; b **←** 1; c **←** 1  **While** c < n **execute**  c **←** c + 1  d **←** 2  **While** c ≤ n **AND** d ≤ a **DIV** 2 **execute**  **If** a **MOD** d **=** 0 **then**  c **←** c + 1  b **←** d  **EndIf**  d **←** d + 1  **EndWhile**  a **←** a + 1  b **←** a  **EndWhile**  **return** b  **EndAlgorithm** |
| C.  **Algorithm** identificare(n):  a **←** 1  b **←** 1  c **←** 1  **While** c < n **execute**  a **←** a + 1  d **←** 2  **While** c < n **AND** d ≤ a **execute**  **If** a **MOD** d **=** 0 **then**  c **←** c + 1  b **←** d  **EndIf**  d **←** d + 1  **EndWhile**  **EndWhile**  **return** b  **EndAlgorithm** | D.  **Algorithm** identificare(n):  a **←** 1; b **←** 1; c **←** 1  **While** c < n **execute**  a **←** a + 1  b **←** a  c **←** c + 1  d **←** 2  f **←** false  **While** c ≤ n **AND** d ≤ a **DIV** 2 **execute**  **If** a **MOD** d = 0 **then**  c **←** c + 1  b **←** d  f **←** true  **EndIf**  d **←** d + 1  **EndWhile**  **If** f **then**  c **←** c - 1  **EndIf**  **EndWhile**  **return** b  **EndAlgorithm** |

**4.** Legyen a factoriPrimi(n, d, k, x) algoritmus, amely meghatározza egy ***n*** szám ***k*** darab törzstényezőjét a ***d*** értékkel kezdődően. ***n***, ***d*** és ***k*** természetes számok, a kimenet, a ***k*** darab törzstényezőt tároló ***x*** sorozat (1 ≤ ***n*** ≤ 10000, 2 ≤ ***d*** ≤ 10000, 0 ≤ ***k*** ≤ 10000).

**Algorithm** factoriPrimi(n, d, k, x):

**If** n **MOD** d = 0 **then**

k ← k + 1

x[k] ← d

**EndIf**

**While** n **MOD** d = 0 **execute**

n ← n **DIV** d

**EndWhile**

**If** n > 1 **then**

factoriPrimi(n, d + 1, k, x)

**EndIf**

**EndAlgorithm**

Állapítsátok meg, hogy hányszor hívja meg önmagát a factoriPrimi(n, d, k, x) algoritmus, amikor a következő programrészletből hívjuk meg:

n ← 120; d ← 2; k ← 0;

factoriPrimi(n, d, k, x)

3-szor

5-ször

9-szer

ugyanannyiszor, mint amikor a következő programrészletből hívjuk meg:

n ← 750; d ← 2; k ← 0;

factoriPrimi(n, d, k, x)

**5.** Adott a generare(a, b, k, x) algoritmus, ahol ***a*** és ***b*** természetes számok (10 ≤ ***a*** ≤ ***b*** ≤ 106), ***k*** természetes szám (1 ≤ ***k*** ≤ ***b*** – ***a***), ***x*** pedig egy ***n*** természetes számot tároló vektor (***x***[1], ***x***[2], …, ***x***[***n***]).

**Algorithm** generare(a, b, k, x):

k ← 0

actual ← 11

ratie ← 11

**While** actual ≤ b **execute**

**If** actual ≥ a **then**

k ← k + 1

x[k] ← actual

**EndIf**

actual ← actual + ratie

**If** actual > 9 \* ratie **then**

actual ← ratie \* 10 + 1

ratie ← actual

**EndIf**

**EndWhile**

**EndAlgorithm**

Melyek igazak a következő állítások közül?

1. Az algoritmus az ***x*** vektorban egy számtani haladványt generál, ahol az első elem ***a***-val egyenlő és a lépték 11.
2. Az algoritmus az ***x*** vektorban azoknak a természetes számoknak a sorozatát generálja, amelyek az [***a***, ***b***] intervallumhoz tartoznak és számjegyeik azonosak.
3. Az algoritmus időbonyolultsága ***O***(***b*** – ***a***).
4. Az algoritmus időbonyolultsága ***O***(log10***b*** – log10***a***).

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

**Algorithm** calcul(n):

x ← 0, z ← 1

**While** z ≤ n **execute**

x ← x + 1

z ← z + 2 \* x

z ← z + 1

**EndWhile**

**return** x

**EndAlgorithm**

Az alábbi állítások közül, melyek **NEM** helyesek?

1. Ha ***n*** < 8, akkor az algoritmus 3-at téríti vissza
2. Ha ***n*** ≥ 85 és ***n*** < 100, akkor az algoritmus 9-et téríti vissza
3. Az algoritmus kiszámítja és visszatéríti az ***n***-nél kisebb négyzetszámok darabszámát
4. Az algoritmus kiszámítja és visszatéríti az ***n*** szám négyzetgyökének egész részét

**7.** Adott a ***mat*** négyzetes mátrix, amelynek ***n*** sora és ***n*** oszlopa van (***n*** páratlan természetes szám, 3 ≤ ***n*** ≤ 100). Adott a puneB(mat, n, i, j) algoritmus, amely a ***mat*** mátrix bizonyos pozícióira 'b' karaktereket ír. Az ***i*** és ***j*** természetes számok 1 ≤ ***i*** ≤ ***n***, 1 ≤ ***j*** ≤ ***n***.

**Algorithm** puneB(mat, n, i, j):

**If** i ≤ n **DIV** 2 **then**

**If** j ≤ n - i **then**

mat[i][j] ← 'b'

puneB(mat, n, i, j + 1)

**else**

puneB(mat, n, i + 1, i + 2)

**EndIf**

**EndIf**

**EndAlgorithm**

Állapítsátok meg, hányszor hívja meg önmagát a puneB(mat, n, i, j) algoritmus, ha a következő programrészletből hívjuk meg:

n ← 7, i ← 2, j ← 4

puneB(mat, n, i, j)

1. 5-ször
2. ugyanannyiszor, mint amikor a következő programrészletből hívjuk meg:

n ← 9, i ← 3, j ← 5

puneB(mat, n, i, j)

1. 10-szer
2. 0-szor

**8.** Adott a calculCuCaractere(s, n, p, q, nr) algoritmus, ahol ***s*** egy ***n*** elemű karakterlánc és ***n*** természetes szám (1 ≤ ***n*** ≤ 9), ***p***, ***q*** és ***nr*** természetes számok (1 ≤ ***p*** ≤ ***n***, 1 ≤ ***q*** ≤ ***n***, ***p*** ≤ ***q***).

**Algorithm** calculCuCaractere(s, n, p, q, nr):

**If** p > q **then**

**return** nr

**else**

**If** s[p] < '0' **OR** s[p] > '9' **then**

**return** nr + calculCuCaractere(s, n, p + 1, q, 0)

**else**

**return** calculCuCaractere(s, n, p + 1, q, 10 \* nr + s[p] - '0')

**Endif**

**EndIf**

**EndAlgorithm**

Állapítsátok meg, hogy a következő állítások közül melyek igazak, miközben ***s***[***p***..***q***]-val jelöljük a karakterlánc azon tömbszakaszát, amely a ***p*** pozíción kezdődik és a ***q*** pozíción ér véget.

A. Az algoritmus kiszámítja és visszatéríti az ***s***[***p***..***q***] tömbszakasz számjegykaraktereinek összegét.

B. Az algoritmus számmá alakítja az ***s***[***p***..***q***] tömbszakasz karaktereit és visszatéríti ezt a számot.

C. Az algoritmus kiszámítja és visszatéríti az ***s***[***p***..***q***] tömbszakaszban található számok összegét.

D. Az algoritmus meghatározza az ***s***[***p***..***q***] tömbszakaszban található számjegyek összegét.

**9.** Egy ***n*** elemű karakterláncnak ***k*** hosszú periódusa van, ha a karakterlánc kialakítható egy ***k*** hosszú karakterlánc ismételt egymásután-ragasztása által (2 ≤ ***n*** ≤ 200, 1 ≤ ***k*** ≤ 100, 2 \* ***k*** ≤ ***n***). Maximális periódusnak hívjuk a karakterlánc leghosszabb periódusát. A következő algoritmus meghatározza az ***n*** elemű ***x*** karakterlánc maximális ***pm*** periódusát, ahol ***n*** természetes szám. Ha az ***x***-nek nincs periódusa, ***pm*** értéke -1.

**Algorithm** perioadaMax(n, x)

pm ← -1

**For** i = n **DIV** 2, 1, -1 **execute**

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

pos ← i

**While** pos < n **execute**

**If** x[pos + 1] ≠ x[pos **MOD** i + 1] **then**

pos ← n + 1

**EndIf**

pos ← pos + 1

**EndWhile**

**If** pos = n **then**

**return** i

**EndIf**

**EndIf**

**EndFor**

**return** -1

**EndAlgorithm**

Állapítsátok meg, hogy a következő állítások közül melyek igazak.

ha ***n*** = 8 és ***x*** = "abababab", akkor ***pm*** = 4.

ha ***n*** = 7 és ***x*** = "abcxabc", akkor ***pm*** = 1.

ha ***n*** = 12 és ***x*** = "abcabcabcabc", akkor ***pm*** = 3.

ha ***n*** = 15 és ***x*** = "abcabcabcabcabc", akkor az algoritmus nem téríti vissza a helyes értéket

**10.** A generare(n) algoritmus az ***n*** természetes számot (0 < ***n*** < 10 000) dolgozza fel.

**Algorithm** generare(n):

nr ← 0

**For** i = 0, 2500 **execute:**

folosit[i] ← ***False***

**EndFor**

**While NOT** folosit[n**] execute:**

suma ← 0

folosit[n] = ***True***

**While** n ≠ 0 **execute**

cifra ← n **MOD** 10

suma ← suma + cifra \* cifra \* cifra

n ← n **DIV** 10

**EndWhile**

n ← suma

nr ← nr + 1

**EndWhile**

**return** nr

**EndAlgorithm**

Határozzátok meg az algoritmus hatását.

1. kiszámítja és visszatéríti az ***n*** szám számjegyei köbének összegét, ameddig az összeg egyenlő lesz ***n***-nel
2. felülírja az ***n*** számot számjegyei köbének összegével
3. kiszámítja az ***n*** szám számjegyei köbének összegét, amíg az összeg egyenlővé válik egy előzőleg kiszámított értékkel és visszatéríti az elvégzett összeadások darabszámát
4. kiszámítja és visszatéríti az ***n*** szám feldolgozásának darabszámát, amíg egy előző lépésben kiszámított értéket kapunk vagy magát az ***n*** számot; feldolgozás alatt az ***n*** szám felülírását értjük számjegyei köbének összegével

**11.** Adott a prelucrează(v, k) algoritmus, ahol ***v*** egy ***k*** elemű, természetes számokból álló sorozat (1 ≤ ***k*** ≤ 1 000).

**Algorithm** prelucrează(v, k)

i ← 1, n ← 0

**While** i ≤ k **AND** vi ≠ 0 **execute**

y ← vi, c ← 0

**While** y > 0 **execute**

**If** y **MOD** 10 > c **then**

c ← y **MOD** 10

**EndIf**

y ← y **DIV** 10

**EndWhile**

n ← n \* 10 + c

i ← i + 1

**EndWhile**

**return** n

**EndAlgorithm**

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

1. ***v*** = (194, 121, 782) és ***k*** = 3
2. ***v*** = (9, 2, 8) és ***k*** = 3
3. ***v*** = (8, 2, 9) és ***k*** = 3
4. ***v*** = (912, 0, 120, 8, 0) és ***k*** = 5

**12.** A conversie(s, lung)algoritmusnak át kellene alakítania az ***s*** karakterláncot, amely egy 16-os számrendszerben megadott számot ábrázol, a megfelelő értékű, 10-es számrendszerben ábrázolt számmá, amelyet 32 biten tárolunk. Az ***s*** karaktereinek darabszáma ***lung***, a karakterek számjegyek és az 'A', 'B', 'C', 'D', 'E', 'F' nagybetűk lehetnek (***lung*** természetes szám, 1 ≤ ***lung*** ≤ 10).

1. **Algorithm** conversie(s, lung):
2. **If** lung > 0 **then**
3. **If** s[lung]≥ 'A' **then**
4. **return** conversie(s, lung - 1) \* 16 + s[lung] - 'A' + 10
5. **else**
6. **return** conversie(s, lung - 1) \* 16 + s[lung]- '0'
7. **EndIf**
8. **else**
9. **return** 0
10. **EndIf**

**11.EndAlgorithm**

Az alábbi állítások közül, melyek **NEM** helyesek?

1. ha ***s*** hossza 1, az algoritmus végtelen ciklusba kerül
2. a 6. sorban levő utasítást ***lung***-szor hajtja végre
3. a 9. sorban levő utasítást egyszer hajtja végre
4. ha ***s*** = "7FFFFFFFF", az algoritmus helytelen értéket térít vissza

**13.** A következő algoritmusok közül melyik számítja ki helyesen ***ab***-nek értékét (***a*** és ***b*** természetes számok, 1 ≤ ***a*** ≤ 11, 1 ≤ ***b*** ≤ 11)?

|  |  |
| --- | --- |
| A.  **Algorithm** expo(a, b):  rezultat ← 1  **While** b > 0 **execute**  **If** b **MOD** 2 = 1) **then**  rezultat ← rezultat \* a  **EndIf**  b ← b **DIV** 2  a ← a \* 2  **EndWhile**  **return** rezultat  **EndAlgorithm** | B.  **Algorithm** expo(a, b):  rezultat ← 1  **While** b > 0 **execute**  **If** b & 1 **then** // AND pe biți  rezultat ← rezultat \* a  **EndIf**  b >>= 1 // translatare la dreapta  a ← a \* a  **EndWhile**  **return** rezultat  **EndAlgorithm** |
| C.  **Algorithm** expo(a, b):  **If** b = 0 **then**  **return** 1  **EndIf**  **return** a \* expo(a, b - 1)  **EndAlgorithm** | D.  **Algorithm** expo(a, b):  **If** b = 0 **then**  **return** 1  **EndIf**  **If** b **MOD** 2 = 0 **then**  **return** expo(a, b **DIV** 2) \* expo(a, b **DIV** 2)  **else**  **return** a \* expo(a, b **DIV** 2) \* expo(a, b **DIV** 2)  **EndIf**  **EndAlgorithm** |

**14.** Az alábbi algoritmusok közül melyik téríti vissza az ***a*** szám legnagyobb többszörösét, amely kisebb vagy egyenlő ***b***-vel (***a*** és ***b*** természetes számok, 0 < ***a*** < 10 000, 0 < ***b*** < 10 000, ***a*** < ***b***)?

|  |  |
| --- | --- |
| A.  **Algorithm** f(a, b):  c ← b  **While** c **MOD** a = 0 **execute:**  c ← c – 1  **EndWhile**  **return** c  **EndAlgorithm** | B.  **Algorithm** f(a, b):  **If** a < b **then**  **return** f(2 \* a, b)  **else**  **If** a = b **then**  **return** a  **else**  **return** b  **EndIf**  **EndIf**  **EndAlgorithm** |
| C.  **Algorithm** f(a, b):  **return** b **DIV** a \* a  **EndAlgorithm** |
| E.  **Algorithm** f(a, b):  **If** b **MOD** a = 0 **then**  **return** b  **EndIf**  **return** f(a, b - 1)  **EndAlgorithm** | D.  **Algorithm** f(a, b):  c ← a  **While** c < b **execute:**  c ← c + a  **EndWhile**  **If** c = b **then**  **return** c  **else**  **return** c – a  **EndIf**  **EndAlgorithm** |

**15.** Adott az aparține(x, a, n) algoritmus, amely vizsgálja, hogy egy természetes ***x*** szám hozzátartozik-e az ***n*** elemű ***a*** halmazhoz; ***a*** egy ***n*** elemű, természetes számokból álló sorozat (1 ≤ ***n*** ≤ 200, 1 ≤ ***x*** ≤ 1000). Továbbá, adott a reuniune(a, n, b, m, c, p), algoritmus, ahol ***a***, ***b*** és ***c*** halmazokat ábrázoló sorozatok, amelyek, rendre ***n***, ***m*** és ***p*** darab természetes számot tárolnak (1 ≤ ***n*** ≤ 200, 1 ≤ ***m*** ≤ 200, 1 ≤ ***p*** ≤ 400).

1. **Algorithm** reuniune(a, n, b, m, c, p):
2. **If** n = 0 **then**
3. **For** i= 1, m **execute**
4. p ← p + 1
5. c[p] ← b[i]
6. **EndFor**
7. **else**
8. **If NOT** aparține(an, b, m) **then**
9. p ← p + 1
10. c[p] ← a[n]
11. **EndIf**
12. reuniune(a, n - 1, b, m, c, p)
13. **EndIf**
14. **EndAlgorithm**

Állapítsátok meg, hogy a következő állítások közül melyek igazak.

1. ha az ***a*** halmaznak 4 eleme van, a reuniune(a, n, b, m, c, p) algoritmus első hívása után a 12. sorban levő utasítás 4-szer hajtódik végre
2. ha az ***a*** halmaznak 5 eleme van, a reuniune(a, n, b, m, c, p)algoritmus első hívása után a 2. sorban levő utasítás 5-ször hajtódik végre
3. ha az ***a*** halmaznak ugyanannyi eleme van, mint a ***b*** halmaznak, a reuniune(a, n, b, m, c, p) algoritmus végrehajtása után a ***c*** halmaznak ugyanannyi eleme lesz, mint az ***a*** halmaznak
4. ha az ***a*** halmaz elemei azonosak a ***b*** halmaz elemeivel, a reuniune(a, n, b, m, c, p) algoritmus végrehajtása után a ***c*** halmaznak ugyanannyi eleme lesz, mint az ***a*** halmaznak

**16.** Egy ***n*** elemű karakterláncot akkor nevezünk*antipalindromnak,* ha a két végponttól azonos távolságra levő minden karakterpár különböző (kivételt képez a középső elem, ha ***n*** páratlan szám). Például, *asdfg* és *xlxe* antipalindromok, de *asdsg* nem az. Legyen az antipalindrom(s, stânga, dreapta) algoritmus, ahol ***s*** egy ***n*** elemű (1 ≤ ***n*** ≤ 10000) karakterlánc (***s***[1], ***s***[2], …, ***s***[***n***]), és ***stânga*** és ***dreapta*** természetes számok. A következő algoritmusok közül melyik térít vissza *True*-t, ha antipalindrom(s, 1, n) alakban hívjuk meg és ***s*** antipalindrom?



**Algorithm** antipalindrom(s, stânga, dreapta):

**If** stânga = dreapta **then**

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

**else**

        prim ← s[stânga]

        ultim ← s[dreapta]

**If** prim = ultim **then**

**return *False***

**else**

**return** antipalindrom(s, stânga + 1, dreapta - 1)

**EndIf**

**EndIf**

**EndAlgorithm**



**Algorithm** antipalindrom(s, stânga, dreapta):

**If** stânga ≥ dreapta **then**

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

**EndIf**

    prim ← s[stânga]; ultim ← s[dreapta]

**If** prim = ultim **then**

**return *False***

**else**

**return** antipalindrom(s, stânga + 1, dreapta - 1)

**EndIf**

**EndAlgorithm**



**Algorithm** antipalindrom(s, stânga, dreapta):

**If** stânga > dreapta **then**

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

**else**

        prim ← s[stânga];ultim ← s[dreapta]

**If** prim ≠ ultim **then**

**return *False***

**else**

**return** antipalindrom(s, stânga + 1, dreapta - 1)

**EndIf**

**EndIf**

**EndAlgorithm**

**Algorithm** antipalindrom(s, stânga, dreapta):

**If** stânga > dreapta **then**

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

**EndIf**

    prim ← s[stânga]**;** ultim ← s[dreapta]

**If** prim ≠ ultim **then**

**return *True***

**EndIf**

**return** antipalindrom(s, stânga + 1, dreapta - 1)

**EndAlgorithm**

**17.** Adott a fibonacci(n) algoritmus, ahol ***n*** természetes szám (1 ≤ ***n*** ≤ 100).

**Algorithm** fibonacci(n):

**If** n ≤ 1 **then**

**return** 1

**else**

**Write** “Bravo”

**return** fibonacci(n - 1) + fibonacci(n - 2)

**EndIf**

**EndAlgorithm**

Hányszor lesz kiírva a “Bravo” szöveg, ha fibonacci(n) alakban hívjuk?

1. fibonacci(n)-szer
2. fibonacci(n – 1)-szer
3. (fibonacci(n) – 1)-szer
4. (fibonacci(n) – fibonacci(n - 1))-szer

**18.** Adott a következő kifejezés: ***E***(***x***) *=* ***a***0 *+* ***a***1 *\*****x*** *+* ***a***2 *\** ***x***2 *+* ***a***3 *\** ***x***3 *+* ***a***5 *\** ***x***5, ahol ***a***0, ***a***1,***a***2,***a***3, ***a***5 és az ***x*** nullától különböző valós szám. Az ***a*** tömb elemei szintén valós számok. Ha az ***E*(*x*)** kifejezés értékét minimális darabszámú szorzással számítjuk ki, akkor a szorzások darabszáma:

A. 4 B. 5 C. 7 D. 11

**19.** Legyen ***bnbn* - 1...*b*0** a ***B*** természetes szám alakja a kettes számrendszerben (2021 ≤ ***B*** ≤ 20212021). Melyek igazak a következő állítások közül?

1. Ha a ***b*0** – ***b*1** + ***b*2** – ***b*3** + ... + (-1)***n*** \* ***bn*** kifejezés értéke nulla, akkor ***B*** osztható 3-mal
2. Ha a ***b*0** – ***b*1** + ***b*2** – ***b*3** + ... + (-1)***n*** \* ***bn*** kifejezés értéke osztható 3-mal, akkor ***B*** osztható 3-mal
3. ***B*** osztható 3-mal, ha a bináris számjegyeinek összege osztható 3-mal, de nem osztható 9-cel
4. Ha ***B*** osztható 3-mal, akkor a ***b*0** – ***b*1** + ***b*2** – ***b*3** + ... + (-1)***n*** \* ***bn*** kifejezés értéke osztható 3-mal

**20.** Szeptember 15.-én az iskola igazgatója felkéri az első osztályos gyermekeket, hogy forduljanak arccal feléje, majd abból a célból, hogy felvezethesse őket az osztálytermekbe, kiadja a parancsot: „Balra át!” A gyermekek bizonytalanok. Van, aki balra fordul, van, aki jobbra. Aki szemben találja magát a mellette állóval, azt hiszi, hogy rosszul fordult és egy időegység alatt egyszer sarkon fordul.

Mivel az igazgató kíváncsi, hogy hány időegység alatt „nyugszik meg” a sor, Pistike írt gyorsan egy programot, amelynek ezt ki kellene számítania. A gyermekek száma ***n*** (1 ≤ ***n*** ≤ 100), a kezdeti konfiguráció pedig egy ***x*** karakterlánc, amelyben az 's' illetve a 'd' betű jelzi, hogy az illető gyermek balra, illetve jobbra fordult. Pistikének nem volt ideje befejezni a programot:

**Algorithm** stângaDreapta(n, x):

stop = ***False***

timp = 0

**While NOT** stop **execute**

i = 0

**stop = *True***

**While** i < n **execute**

**If** x[i] = 'd' **AND** x[i + 1] = 's' **then**

...

**else**

i = i + 1

**EndIf**

**EndWhile**

**If NOT** stop **then**

timp = timp + 1

**EndIf**

**EndWhile**

**return** timp

**EndAlgorithm**

Adjátok meg, hogy mit kellene írni a … helyére ahhoz, hogy a program azoknak az időegységeknek a számát térítse vissza, amelyek eltelnek addig amíg megnyugszik a sor:

1. x[i] = 's'

x[i + 1] = 'd'

i = i + 1

stop = ***False***

1. x[i] = 'd'

x[i + 1] = 's'

i = i + 2

stop = ***False***

1. aux = x[i]

x[i] = x[i + 1]

x[i + 1] = aux

i = i + 1

1. x[i] = 's'

x[i + 1] = 'd'

i = i + 2

stop = ***False***

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

**Algorithm** ceFace(n)

    nr ← 0

**For** d ← 1, n **execute**

**If** n **MOD** d = 0 **then**

            nr ← nr + 1

**EndIf**

**EndFor**

**If** nr = 2 **then**

**return *TRUE***

**else**

**return *FALSE***

**EndIf**

**EndAlgorithm**

Állapítsátok meg, hogy a következő állítások közül melyek igazak.

1. Az algoritmus *True*-t térít vissza, ha ***n*** páratlan szám.
2. Az algoritmus *True*-t térít vissza, ha ***n*** páros szám.
3. Az algoritmus *True*-t térít vissza, ha ***n*** prímszám.
4. Az algoritmus *True*-t térít vissza, ha ***n*** négyzetszám.

**22.** Tudjuk, hogy ***x* < *y*** (***x*** és ***y*** valós számok). Melyik alábbi kifejezés értéke *True* akkor és csakis akkor, ha a ***t*** változó értéke (***t*** valós szám) **NEM** tartozik az (***x***, ***y***) intervallumhoz?

1. (t > x) **OR** (t < y)
2. (t ≤ x) **OR** (t ≥ y)
3. (t ≤ x) **AND** (t ≥ y)
4. (t > x) **AND** (t < y)

**23.** Legyen az f(n) algoritmus, ahol ***n*** természetes szám (1 ≤ ***n*** ≤ 10000):

**Algorithm** f(n):

    r ← 0

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

        r ← r + (n **MOD** 10) \* (n **MOD** 2)

        n ← n **DIV** 10

**EndWhile**

**return** r

**EndAlgorithm**

Mivel kell helyettesítenünk a … -t, ahhoz, hogy a két algoritmus minden esetben ugyanazt az értéket térítse vissza?

**Algorithm** fr(n):

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

**return** ...

**EndIf**

**return** 0

**EndAlgorithm**

1. (n **MOD** 2) \* (n **MOD** 10) + fr(n **DIV** 10)
2. (n **MOD** 2) \* (n **MOD** 10) \* fr(n **DIV** 10)
3. (n **MOD** 10) + fr(n **DIV** 10)
4. (n **MOD** 2) \* (n **MOD** 10) + fr(n **MOD** 10)

**24.** Abbóla célból, hogy olyan ***n*** számjegyű számokat generáljunk, amelyek csak 0, 2, 9számjegyekből állnak egy olyan algoritmust alkalmazunk, amely ***n*** = 2 esetében a 20, 22, 29, 90, 92, 99 számokat generálja növekvő sorrendben. Ha ***n*** = 4és ugyanazt az algoritmust alkalmazzuk, melyik szám következik a 2009 után?

1. 2022
2. 2090
3. 2010
4. Egyik sem a fentiek közül

**25.** A következő algoritmus az ***n*** (2 ≤ ***n*** ≤ 10000) természetes számot tároló ***v*** vektorral dolgozik (***v***[1], ***v***[2], …, ***v***[***n***]). A **/** operátor a valós osztást jelzi (például: 3 / 2 = 1,5). A ***v*** vektor tartalmaz legalább egy páros és legalább egy páratlan számot.

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

    a ← 0

    b ← 0

**For** i ← 1, n **execute**

**If** v[i] **MOD** 2 = 0 **then**

            a ← a + v[i]

            b ← b + 1

**EndIf**

**EndFor**

**return** a / b

**EndAlgorithm**

Állapítsátok meg, hogy a következő állítások közül melyek igazak.

1. Az algoritmus kiszámítja és visszatéríti a ***v*** vektorban található páros számok darabszámát
2. Az algoritmus kiszámítja és visszatéríti a ***v*** vektoran található páros számok átlagát
3. Az algoritmus kiszámítja és visszatéríti a ***v*** vektoran található páros számok összegét
4. Az algoritmus kiszámítja és visszatéríti a ***v*** vektoran található páratlan számok átlagát

**26.** Az alábbi algoritmusban ***v*** egy ***n*** elemű, természetes számokból álló vektor (***v***[1], ***v***[2], …, ***v***[***n***]) ***n*** természetes szám (1 ≤ ***n*** ≤ 10000).

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

    a ← 0

**For** i ← 1, n **execute**

        ok ← 1

        b ← v[i]

**While** (b ≠ 0) **AND** (ok = 1) **execute**

**If** b **MOD** 2 = 0 **then**

                ok ← 0

**EndIf**

            b ← b **DIV** 10

**EndWhile**

**If** ok = 1 **then**

            a ← a + v[i]

**EndIf**

**EndFor**

**return** a

**EndAlgorithm**

Állapítsátok meg, hogy a következő állítások közül melyek igazak.

1. Az algoritmus kiszámítja és visszatéríti a ***v*** vektor páratlan elemeinek összegét
2. Az algoritmus kiszámítja és visszatéríti a ***v*** vektor azon elemeinek összegét, amelyek kettőhatványok
3. Az algoritmus kiszámítja és visszatéríti a ***v*** vektor azon elemeinek összegét, amelyeknek konfigurációjában csak páros számjegyek szerepelnek
4. Az algoritmus kiszámítja és téríti vissza összege a ***v*** vektor azon elemeinek összegét, amelyeknek konfigurációjában csak páratlan számjegyek szerepelnek

**27.** Adjátok meg, hogy a következő algoritmusok közül melyik számítja ki helyesen egy egész szám moduluszát (abszolút értékét). Egy logikai kifejezés értéke 1, ha igaz, és 0 ha hamis.



**Algorithm** modul(n):

**return** n \* (-2 \* (n < 0) + 1)

**EndAlgorithm**



**Algorithm** modul(n):

**If** n < 0 **then**

**return** n \* (-1)

**EndIf**

**return** n

**EndAlgorithm**

**Algorithm** modul(n):

**If** n < 0 **then**

**return** n \* (-1)

**else**

**return** n

**EndIf**

**EndAlgorithm**

**Algorithm** modul(n):

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

**return** n \* (-1)

**else**

**return** n

**EndIf**

**EndAlgorithm**

**28.** Mi lesz az alábbi kifejezés értéke, ha ***x*** = 15 és ***y*** = 17?

(**NOT** (x **MOD** 10 = 0)) **AND (**y **MOD** 2 = 0) **AND** (x < y)

1. *True*
2. *False*
3. Hiba
4. A kifejezést nem lehet kiértékelni

**29.** Adott a ceFace(n, i) algoritmus, ahol ***n*** természetes szám (2 ≤ ***n*** ≤ 1000).

**Algorithm** ceFace(n, i):

**If** i = 1 **then**

**return** i

**else**

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

**return** i + ceFace(n, i - 1)

**else**

**return** ceFace(n, i - 1)

**EndIf**

**EndIf**

**EndAlgorithm**

Állapítsátok meg, hogy a következő állítások közül melyek igazak a ceFace(n, n)meghívásának esetében?

1. Az algoritmus meghatározza és visszatéríti az ***n*** szám legnagyobb osztójának rákövetkezőjét
2. Az algoritmus kiszámítja és visszatéríti azon számok összegét, amelyek nem prímek és kisebbek vagy egyenlők, mint ***n***
3. Az algoritmus kiszámítja és visszatéríti az ***n*** szám valódi osztóinak összegét
4. Az algoritmus kiszámítja és visszatéríti az ***n*** szám valódi és nem valódi osztóinak összegét

**30.** A magic(s, n) algoritmus paraméterei az  ***n*** elemű ***s*** karakterlánc (***s***[1], ***s***[2], …, ***s***[***n***]) és az ***n*** természetes szám (1 ≤ ***n*** ≤ 10000).

**Algorithm** magic(s, n):

   i ← 1

   f ← 1

**While** i ≤ n **DIV** 2 **execute**

**If** s[i] ≠ s[n - i + 1] **then**

           f ← 0

**EndIf**

       i ← i + 1

**EndWhile**

**return** f

**EndAlgorithm**

Állapítsátok meg, hogy a következő állítások közül melyek igazak:

1. Az algoritmus 1-et térít vissza, ha ***s*** karakterinek darabszáma páros.
2. Az algoritmus 1-et térít vissza, ha ***s*** karakterinek darabszáma páratlan.
3. Az algoritmus 1-et térít vissza, ha ***s*** palindrom.
4. Az algoritmus 1-et térít vissza, ha ***s*** karakterei különbözők.