UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

MATEK-INFO verseny – 2025 április 12

Informatika írásbeli

FONTOS MEGJEGYZÉS:

Más pontosítások hiányában:

- Az aritmetikai műveleteket végtelen adattípusokon végezzük (nincs túlcsordulás és alulcsordulás).
- Minden vektort, mátrixot és karakterláncot 1-től sorszámozunk (indexelünk).
- Az aktuális paraméterek értékeire vonatkozó megszorítások a kezdeti hívás pillanatában érvényesek.
- Egy vektor tömbszakaszát a vektor olyan elemei alkotják, amelyek egymás utáni pozíciókon találhatók.
- Ha ugyanabban a sorban több egymásutáni értékadó utasítás található, ezek "; "-vel vannak elválasztva.

1. Adott a `calcul(n, c1, c2)` algoritmus, ahol $\boldsymbol{n}$ egy természetes szám ($1 \leq \boldsymbol{n} \leq 10^4$) és $\boldsymbol{c1}$, $\boldsymbol{c2}$ számjegyek ($0 \leq \boldsymbol{c1}, \boldsymbol{c2} \leq 9$).

```
Algorithm calcul(n, c1, c2):
    If n = 0 then
        Return 0
    EndIf
    If n MOD 10 = c1 then
        Return calcul(n DIV 10, c1, c2) * 10 + c2
    Else
        Return calcul(n DIV 10, c1, c2) * 10 + n MOD 10
    EndIf
EndAlgorithm
```

Mit térít vissza az algoritmus, ha $\boldsymbol{n} = 1999$, $\boldsymbol{c1} = 1$ és $\boldsymbol{c2} = 0$?

A. 1000
B. 999
C. 1099
D. 1990

2. Adott a `ceFace(m, n)` algoritmus, ahol $\boldsymbol{m}$ és $\boldsymbol{n}$ természetes számok ($1 \leq \boldsymbol{m}, \boldsymbol{n} \leq 100$):

```
 1. Algorithm ceFace(m, n):
 2.    c ← 1; i ← n
 3.    While i > 0 execute
 4.        If i MOD 2 = 1 then
 5.            c ← c * m
 6.            i ← i - 1
 7.        Else
 8.            m ← m * m
 9.            i ← i DIV 2
10.        EndIf
11.    EndWhile
12.    Return c
13. EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Ha az algoritmust `ceFace(2, 5)` alakban hívjuk meg, a visszatérített érték 30.
B. Ha a `ceFace(m, n)` hívás eredményeképpen a visszatérített érték $\boldsymbol{x}$, nem létezik más olyan $\boldsymbol{m1}$, $\boldsymbol{n1}$ ($\boldsymbol{m1} \neq \boldsymbol{m}$ és $\boldsymbol{n1} \neq \boldsymbol{n}$) értékpár, melyre a `ceFace(m1, n1)` hívás eredménye ugyanaz az $\boldsymbol{x}$ érték.
C. $\boldsymbol{n}$ egyedüli értéke, melyre a 6. sor 2-szer hajtódik végre a `ceFace(m, n)` hívás eredményeképpen, 5.
D. Ha az algoritmust `ceFace(5, 8)` alakban hívjuk meg, a 6. sor egyszer hajtódik végre.

3. Adott a `ceFace(b, n, a)` algoritmus, ahol $\boldsymbol{b}$ és $\boldsymbol{n}$ természetes számok ($2 \leq \boldsymbol{b}, \boldsymbol{n} \leq 100$) és $\boldsymbol{a}$ egy $\boldsymbol{n}$ természetes számból álló vektor ($\boldsymbol{a}[1], \boldsymbol{a}[2], ..., \boldsymbol{a}[\boldsymbol{n}]$, $0 \leq \boldsymbol{a}[\boldsymbol{i}] < \boldsymbol{b}$, ahol $\boldsymbol{i} = 2, 3, ..., \boldsymbol{n}$ és $0 < \boldsymbol{a}[1] < \boldsymbol{b}$):

```
Algorithm ceFace(b, n, a):
    v ← a[1]
    For i ← 2, n execute
        v ← v * b + a[i]
    EndFor
    Return v
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. A `ceFace(2,6,[1,0,1,0,1,1])` hívás eredménye 43.
B. A `ceFace(9,3,[7,6,5])` hívás eredménye 626.
C. Ha $\boldsymbol{a}[\boldsymbol{n}] = 0$, a `ceFace(b, n, a)` hívás páros számot térít vissza.
D. Ha $\boldsymbol{b1} > \boldsymbol{b2}$, akkor a `ceFace(b1, n, a)` hívás nagyobb számot térít vissza, mint a `ceFace(b2, n, a)` hívás.

4. Adott az $\boldsymbol{n}$ egész szám ($-100 \leq \boldsymbol{n} \leq 100$).

A következő kifejezések közül melyek értéke akkor és csakis akkor *True*, ha $\boldsymbol{n}$ **NEM** tartozik a $\{-8\} \cup \{-4, -3, ..., 8\}$ halmazhoz?

A. `(n ≤ -8) AND (n ≥ -8) AND (n ≤ -4) AND (n ≥ 8)`
B. `(n < -8) OR ((n > -8) AND (n < -4)) OR (n > 8)`
C. `(n < -8) OR ((n > -8) OR (n < -4)) AND (n > 8)`
D. `((n < -4) AND (n ≠ -8)) OR (n > 8)`

5. Adott a `cautBin(st, dr, y, x, n)` algoritmus, ahol ***st*** és ***dr*** természetes számok, ***x*** egy ***n*** egész számból álló növekvően rendezett vektor ($1 \le$ ***st***, ***dr***, ***n*** $\le 10^4$, ***x***[1], ***x***[2], ..., ***x***[***n***], $-10^3 \le$ ***x***[***i***] $\le 10^3$ ahol ***i*** = 1, 2, ..., ***n***) és ***y*** egy egész szám ($-10^3 \le$ ***y*** $\le 10^3$, ***x***[1] < ***y***) mely nem szerepel a vektorban. Az algoritmust a következő alakban hívjuk meg: `cautBin(1, n, y, x, n)`.

```
Algorithm cautBin(st, dr, y, x, n):
    If st < dr then
        mij ← (st + dr) DIV 2
        If y < x[mij] then
            Return cautBin(st, mij, y, x, n)
        Else
            Return cautBin(mij + 1, dr, y, x, n)
        EndIf
    Else
        .........
    EndIf
EndAlgorithm
```

Mivel kell helyettesíteni a pontozott részt ahhoz, hogy az algoritmus visszatérítse az első olyan elem pozícióját az ***x*** vektorból, amely nagyobb, mint ***y***. Ha nem létezik ilyen szám, a visszatérített érték -1.

A.
```
If y > x[dr] then
    Return dr + 1
Else
    Return -1
EndIf
```

B.
```
If y < x[dr] then
    Return dr
Else
    Return -1
EndIf
```

C.
```
If y > x[st] then
    Return st + 1
Else
    Return -1
EndIf
```

D.
```
If y < x[st] then
    Return st
Else
    Return -1
EndIf
```

6. Adott a `calculeaza(x, n)` algoritmus, ahol ***n*** egy természetes szám ($1 \le$ ***n*** $\le 10^4$) és ***x*** egy ***n*** egész számból álló vektor (***x***[1], ***x***[2], ..., ***x***[***n***], $-100 \le$ ***x***[***i***] ≤ 100, ahol ***i*** = 1, 2, ..., ***n***).

```
Algorithm calculeaza(x, n):
    If n MOD 2 = 1 then
        s ← x[n]
    Else
        s ← 0
    EndIf
    For i ← 1, n - 2, 2 execute
        s ← s + x[i] + x[i + 1]
    EndFor
    Return s
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. A `calculeaza([3, -8, -2, 15, -1, 0, 3, 1, 3], 9)` hívás eredményeképpen az algoritmus által visszatérített érték 11.
B. A `calculeaza([2, -1, 7, 5, -9, 0, 3, 1, 12], 9)` hívás eredményeképpen az algoritmus által visszatérített érték 4.
C. A `calculeaza([10, 2, 5, 78, 23, 4, 11], 7)` hívás eredményeképpen az algoritmus által visszatérített érték 133.
D. A `calculeaza([-3, 8, -2, 15, -1, 10], 6)` hívás eredményeképpen az algoritmus által visszatérített érték 27.

7. Adott az `f(n, a, p)` algoritmus, ahol ***n*** és ***p*** természetes számok ($1 \le$ ***n***, ***p*** $\le 10^5$) és ***a*** egy ***n*** számjegyet tartalmazó vektor (***a***[1], ***a***[2], ..., ***a***[***n***], $0 \le$ ***a***[***i***] ≤ 9, ahol ***i*** = 1, 2, ..., ***n***), melyben legalább egy számjegy különbözik 0-tól:

```
Algorithm f(n, a, p):
    s ← 0
    For i ← 1, n execute
       s ← s + a[i]
    EndFor
    For i ← 1, p execute
       If s MOD 3 = 0 then
          s ← s DIV 3
       Else
          Return False
       EndIf
    EndFor
    Return True
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Az algoritmus által visszatérített érték akkor és csakis akkor *True*, ha az ***a*** vektor elemeinek összege 3^p többszöröse.
B. Az algoritmus által visszatérített érték akkor és csakis akkor *True*, ha az ***a*** vektor elemeinek összege 3 hatványa.
C. Az algoritmus által visszatérített érték akkor és csakis akkor *False*, ha az ***a*** vektor elemeinek összege nem osztható 3-mal.
D. Az `f(6, [9, 1, 8, 8, 4, 6], 2)` hívás eredményeképpen az algoritmus által visszatérített érték *True*.

8. Egy ***n*** csúcsú és ***p*** ($0 < p \le n$) összefüggő komponensből álló irányítatlan gráf élei számának maximuma:

A. $\frac{(n-p)\times(n-p+1)}{2}$ B. $(n-p)\times(n-p+1)$ C. $\frac{(n-p)\times(n-p+1)}{4}$ D. $\frac{(n-p)\times(n+p+1)}{2}$

9. Adott az $\boldsymbol{n}$ természetes szám ($10 \leq \boldsymbol{n} \leq 10^4$).

Az `f(n)` algoritmus alábbi változatai közül, melyik téríti vissza $\boldsymbol{n}$ tükörképét?

A.

```
Algorithm f(n):
    If n > 0 then
        Return n MOD 10 + 10 * f(n DIV 10)
    EndIf
    Return 0
EndAlgorithm
```

B.

```
Algorithm f1(n, ogl):
    If n > 0 then
        Return f1(n DIV 10, n MOD 10 + 10 * ogl)
    EndIf
    Return ogl
EndAlgorithm

Algorithm f(n):
    Return f1(n, 0)
EndAlgorithm
```

C.

```
Algorithm f(n):
    ogl ← 0
    While n > 0 execute
        ogl ← (n MOD 10) * 10 + ogl
        n ← n DIV 10
    EndWhile
    Return ogl
EndAlgorithm
```

D.

```
Algorithm f(n):
    ogl ← 0
    While n > 0 execute
        ogl ← ogl * 10 + n MOD 10
        n ← n DIV 10
    EndWhile
    Return ogl
EndAlgorithm
```

10. Adott a `ceFace(x1, y1, x2, y2, x3, y3)` algoritmus, ahol (***x*1**, ***y*1**), (***x*2**, ***y*2**) és (***x*3**, ***y*3**) három különböző mértani pont koordinátái.

```
Algorithm ceFace(x1, y1, x2, y2, x3, y3):
    t ← x1 * (y2 - y3)
    v ← x2 * (y1 - y3)
    z ← x3 * (y1 - y2)
    Return (t - v + z) ≠ 0
EndAlgorithm
```

A következő állítások közül melyek igazak a `ceFace(x1, y1, x2, y2, x3, y3)` hívás esetén?

A. A visszatérített érték *True*, ha az adott pontok egy nem elfajult háromszöget alkotnak.
B. A visszatérített érték *False*, ha az adott pontok kollineárisak.
C. A visszatérített érték *False*, ha az adott pontok egy nem elfajult háromszöget alkotnak.
D. A visszatérített érték *True*, ha az adott pontok kollineárisak.

11. Adott a `h(A, n)` algoritmus, ahol $\boldsymbol{n}$ egy természetes szám ($1 \leq \boldsymbol{n} \leq 10^3$) és $\boldsymbol{A}$ egy $\boldsymbol{n}$ egész számot tartalmazó vektor ($\boldsymbol{A}[1], \boldsymbol{A}[2], ..., \boldsymbol{A}[\boldsymbol{n}]$, és $0 \leq \boldsymbol{A}[\boldsymbol{i}] \leq 100$, ahol $\boldsymbol{i} = 1, 2, ..., \boldsymbol{n}$):

```
Algorithm h(A, n):
    If n = 0 then
        Return 0
    EndIf
    Return h(A, n - 1) + (A[n] MOD 2) * (A[n] MOD 10) * (n MOD 2)
EndAlgorithm
```

Melyik hívás visszatérítési értéke 0?

A. `h([25, 14, 35, 26, 2, 10], 6)`
B. `h([14, 25, 26, 2, 10, 35], 6)`
C. `h([12, 5, 22, 4, 32, 8, 46, 9, 54, 3], 10)`
D. `h([3, 4, 7], 3)`

12. Adott a `ceFace(n)` algoritmus, ahol $\boldsymbol{n}$ egy természetes szám ($0 \leq \boldsymbol{n} \leq 10$).

```
 1. Algorithm ceFace(n):
 2.     e ← 1
 3.     For f ← 1, n execute
 4.         s ← 0
 5.         For j ← 1, f execute
 6.             s ← s + j
 7.         EndFor
 8.         e ← e * s
 9.     EndFor
10.     Return e
11. EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Ha az algoritmust `ceFace(5)` alakban hívjuk meg, a visszatérített érték 2700.
B. Függetlenül $\boldsymbol{n}$ értékétől, a `ceFace(n)` algoritmus soha nem térít vissza 0-át.
C. A `ceFace(9)` hívás eredményének végén ugyanannyi nullás van, mint a `ceFace(10)` hívás eredményének a végén.
D. Ha az algoritmust `ceFace(10)` alakban hívjuk meg, a 6. sor 45-ször hajtódik végre.

13. Adott a `ceva(n)` algoritmus, ahol ***n*** természetes szám ($1 \leq n \leq 10^9$).

```
Algorithm aux(n):
    v1 ← 1
    v2 ← 1
    While v1 < n execute
        v3 ← v1 + v2
        v1 ← v2
        v2 ← v3
    EndWhile
    Return v1 = n
EndAlgorithm
```

```
Algorithm ceva(n):
    If aux(n) then
        Return True
    EndIf
    p ← 10
    gata ← False
    While (n DIV p ≠ 0) AND (NOT gata) execute
        nr1 ← n MOD p
        nr2 ← (n – nr1) DIV p
        If aux(nr1) then
            gata ← ceva(nr2)
        EndIf
        p ← p * 10
    EndWhile
    Return gata
EndAlgorithm
```

Feltételezve, hogy a *Fibonacci* sorozat első hat eleme 1, 1, 2, 3, 5, 8, a következő állítások közül melyek igazak?

A. A `ceva(n)` algoritmus akkor és csakis akkor térít vissza *True*-t, ha ***n*** *Fibonacci-szám*.
B. A `ceva(n)` algoritmus ellenőrzi, hogy ***n*** felírható-e *Fibonacci-számok* összegeként.
C. A `ceva(n)` algoritmus ellenőrzi, hogy ***n*** felírható-e *Fibonacci-számok* szorzataként.
D. Ha ***n*** = 1234589, akkor a `ceva(n)` algoritmus *True*-t térít vissza.

14. Adott a `ceFace(n, f, p)` algoritmus, ahol ***n*** egy természetes szám ($0 \leq n \leq 10^{10}$), ***p*** egy természetes szám ($0 \leq p \leq 100$) és ***f*** egy egész szám ($-1 \leq f \leq 1$).

```
Algorithm ceFace(n, f, p):
    If n = 0 then
        Return f = 1
    EndIf
    c ← n MOD 10
    n ← n DIV 10
    If f = -1 then
        If c < p then
            Return ceFace(n, 0, c)
        Else
            Return False
        EndIf
    EndIf
    If f = 0 then
        If c < p then
            Return ceFace(n, 0, c)
        Else
            If c > p then
                Return ceFace(n, 1, c)
            Else
                Return False
            EndIf
        EndIf
    EndIf
    If f = 1 then
        If c > p then
            Return ceFace(n, 1, c)
        Else
            Return False
        EndIf
    EndIf
EndAlgorithm
```

Az alábbi állítások közül melyek igazak a `ceFace(n DIV 10, -1, n MOD 10)` hívás eredményére vonatkozóan?

A. Bármely ***n*** < 101 értékre a visszatérített érték *False*.
B. Ha ***n*** = 8976532014, a visszatérített érték *True*.
C. Ha ***n*** legkevesebb két egyenlő számjegyet tartalmaz, a visszatérített érték *False*.
D. Ha ***n*** nem tartalmazza a 0-ás számjegyet és a visszatérített érték *True*, akkor a hívás eredménye *True* ***n*** tükörképére is.

15. Adott a következő három algoritmus, mely egy ***n*** természetes szám ($1 \le \boldsymbol{n} \le 10^{12}$) leggyakoribb számjegyét határozza meg: `cifreA(n)`, `cifreB(n)` és `cifreC(n)`.

```
Algorithm cifreA(n):
    c ← n
    maxf ← -1; maxd ← -1
    While c > 0 execute    // (*)
        d ← c MOD 10
        copie ← n
        cnt ← 0
        While copie > 0 execute
            If copie MOD 10 = d then
                cnt ← cnt + 1
            EndIf
            copie ← copie DIV 10
        EndWhile
        If cnt > maxf then
            maxf ← cnt
            maxd ← d
        EndIf
        c ← c DIV 10
    EndWhile
    Return maxd
EndAlgorithm
```

```
Algorithm cifreB(n):
    maxf ← -1
    maxd ← -1
    For i ← 0, 9 execute    // (*)
        c ← n
        cnt ← 0
        While c > 0 execute
            If c MOD 10 = i then
                cnt ← cnt + 1
            EndIf
            c ← c DIV 10
        EndWhile
        If cnt > maxf then
            maxf ← cnt
            maxd ← i
        EndIf
    EndFor
    Return maxd
EndAlgorithm
```

```
Algorithm cifreC(n):
    maxf ← -1; maxd ← -1
    For i ← 9, 0, -1 execute
        c ← n; cnt ← 0
        While c > 0 execute
            If c MOD 10 = i then
                cnt ← cnt + 1
            EndIf
            c ← c DIV 10
        EndWhile
        If cnt > maxf then
            maxf ← cnt
            maxd ← i
        EndIf
    EndFor
    Return maxd
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. `cifreA(123453) = cifreB(123453) = cifreC(123453)`
B. `cifreA(123456) = cifreB(123456) = cifreC(123456)`
C. Létezik legalább egy ***n*** természetes szám, amelyre a három algoritmus három különböző értéket térít vissza.
D. Minden természetes ***n*** számra, a (*)-gal jelölt **`While`** ciklus a `cifreA(n)` algoritmusban kevesebbszer hajtódik végre, mint a (*)-gal jelölt **`For`** ciklus a `cifreB(n)` algoritmusból.

16. Adott a `getSomeMax(n, x)` algoritmus, ahol ***n*** egy természetes szám ($1 \le \boldsymbol{n} \le 10^3$) és ***x*** egy ***n*** egész számból álló vektor (***x***[1], ***x***[2], ..., ***x***[***n***], ahol $-10^3 \le \boldsymbol{x}[\boldsymbol{i}] \le 10^3$, ha ***i*** = 1, 2, ..., ***n***). A `zero(k)` algoritmus egy ***k*** elemű vektort térít vissza, ahol minden elem egyenlő nullával.

```
Algorithm getSomeMax(n, x):
    y ← zero(n + 1)
    For i ← 1, n execute
        y[i + 1] ← y[i] + x[i]
    EndFor
    sm ← y[2]
    For i ← 2, n execute
        For j ← i, n execute
            s ← y[j] - y[i - 1]
            If s > sm then
                sm ← s
            EndIf
        EndFor
    EndFor
    Return sm
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Ha ***n*** = 1, a `getSomeMax(n, x)` algoritmus által visszatérített érték ***x***[1].
B. A `getSomeMax(8, [5, 7, -4, 6, -3, -2, 6, -7])` hívás esetén az algoritmus által visszatérített érték 10.
C. Ha ***n*** = 100 és ***x*** = `[1, 2, 3, ..., 99, 100]`, a `getSomeMax(n, x)` algoritmus által visszatérített érték 4950.
D. Ha az ***x*** vektor minden eleme szigorúan negatív, a `getSomeMax(n, x)` algoritmus a vektor legnagyobb elemét téríti vissza.

17. Adott az `afla(n, x)` algoritmus, ahol $\boldsymbol{n}$ egy természetes szám ($3 \le \boldsymbol{n} \le 10^4$) és $\boldsymbol{x}$ egy $\boldsymbol{n}$ egész számot tartalmazó vektor ($\boldsymbol{x}[1], \boldsymbol{x}[2], ..., \boldsymbol{x}[\boldsymbol{n}]$, $-100 \le \boldsymbol{x}[\boldsymbol{i}] \le 100$, ahol $\boldsymbol{i} = 1, 2, ..., \boldsymbol{n}$):

```
1. Algorithm afla(n, x):
2.     M1 ← x[1]; M2 ← x[2]; M3 ← x[3]
3.     For i ← 1, n execute
4.         If x[i] > M1 then
5.             M3 ← M2
6.             M2 ← M1
7.             M1 ← x[i]
8.         Else
9.             If x[i] > M2 then
10.                M3 ← M2
11.                M2 ← x[i]
12.            Else
13.                If x[i] > M3 then
14.                    M3 ← x[i]
15.                EndIf
16.            EndIf
17.        EndIf
18.    EndFor
19.    Return M1, M2, M3
20.EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Ha az algoritmust `afla(6, [1, 2, 3, 4, 5, 6])` alakban hívjuk meg a visszatérített értékek 6, 5, 4.

B. Ha a 8. és 12. sorok tartalma **`EndIf`** lenne és a 16. és 17. sorokat törölnénk, az algoritmus ugyanazt az eredményt térítené vissza, mint az eredeti algoritmus.

C. Ha $\boldsymbol{M1}$, $\boldsymbol{M2}$ és $\boldsymbol{M3}$ kezdeti értéke $\boldsymbol{x}[3]$, $\boldsymbol{x}[2]$, valamint $\boldsymbol{x}[1]$ lenne, az algoritmus ugyanazt az eredményt térítené vissza, mint az eredeti algoritmus.

D. Ha a 3. sorban `For i ← 1, n execute` helyett `For i ← 4, n execute` lenne, az algoritmus ugyanazt az eredményt térítené vissza, mint az eredeti algoritmus.

18. Adott a `ceFace(A, n)` algoritmus, ahol $\boldsymbol{n}$ egy természetes szám ($1 \le \boldsymbol{n} \le 20$) és $\boldsymbol{A}$ egy $\boldsymbol{n}$ soros és $\boldsymbol{n}$ oszlopos négyzetes mátrix, amely természetes számokat tartalmaz: ($\boldsymbol{A}[1][1], \boldsymbol{A}[1][2], ..., \boldsymbol{A}[\boldsymbol{n}][\boldsymbol{n}]$, és $0 \le \boldsymbol{A}[\boldsymbol{i}][\boldsymbol{j}] \le 200$, ahol $\boldsymbol{i} = 1, 2, ..., \boldsymbol{n}$ és $\boldsymbol{j} = 1, 2, ..., \boldsymbol{n}$).

```
Algorithm ceFace(A, n):
    // Început partea 1
    For i ← 1, n execute
        For j ← i + 1, n execute
            temp ← A[i][j]
            A[i][j] ← A[j][i]
            A[j][i] ← temp
        EndFor
    EndFor
    // Sfârșit partea 1
    // Început partea 2
    For i ← 1, n execute
        For j ← 1, n DIV 2 execute
            temp ← A[i][j]
            A[i][j] ← A[i][n - j + 1]
            A[i][n - j + 1] ← temp
        EndFor
    EndFor
    // Sfârșit partea 2
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. A `ceFace(A, 3)` algoritmus végrehajtása után, az $\boldsymbol{A} = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$, mátrixból $\begin{matrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{matrix}$ lesz.

B. Ha a bemeneti $\boldsymbol{A}$ mátrix a 3-as rendű egységmátrix, akkor ez nem fog módosulni a `ceFace(A, 3)` algoritmus végrehajtása után.

C. A `ceFace(A, n)` algoritmus 90^o-al jobbra forgatja az adott mátrixot, melyet megfelelően módosít.

D. Ha az algoritmusban felcseréljük az `Început partea 1` és `Sfârșit partea 1` közötti részt az `Început partea 2` és `Sfârșit partea 2` közötti résszel, a `ceFace(A, n)` algoritmus ugyanazt az eredményt téríti vissza, mint az eredeti algoritmus.

19. Adott a `ceFace(n)` algoritmus, ahol $\boldsymbol{n}$ egy természetes szám ($0 \le \boldsymbol{n} \le 200$).

```
1. Algorithm ceFace(n):
2.    e ← 0
3.    For i ← 1, n execute
4.        If i MOD 2 = 0 then
5.            e ← e - 2 * i * i
6.        Else
7.            e ← e + 2 * i * i
8.        EndIf
9.    EndFor
10.   Return e
11. EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Bármilyen $\boldsymbol{n}$ páros számra, az algoritmus negatív értéket térít vissza.

B. Az algoritmus kiszámolja az alábbi kifejezés értékét
$0 + 1 * 2 - 2 * 4 + 3 * 6 - 4 * 8 + \ldots + (-1)^{\boldsymbol{n}-1} * \boldsymbol{n} * 2 * \boldsymbol{n}$

C. Ha a `ceFace(n)` algoritmus negatív értéket térít vissza, $\boldsymbol{n}$ egy páros szám.

D. Egyetlen olyan $\boldsymbol{n}$ érték létezik, amelyre a 7. sor pontosan 7-szer hajtódik végre.

20. Adott a `ceFace(n, x)` algoritmus, ahol ***n*** egy természetes szám ($2 \le \boldsymbol{n} \le 10^3$) és ***x*** egy ***n*** egész számot tartalmazó vektor (***x***[1], ***x***[2], ..., ***x***[***n***], -100 ≤ ***x***[***i***] ≤ 100, ahol ***i*** = 1, 2, ..., ***n***). A `zero(k)` algoritmus egy ***k*** elemű vektort térít vissza, ahol minden elem egyenlő nullával. A `minim(n, x)`, illetve a `maxim(n, x)` algoritmusok visszatérítik a legkisebb és a legnagyobb értéket az ***n*** elemű ***x*** vektorból.

```
01. Algorithm ceFace(n, x):
02.     min ← minim(n, x)
03.     max ← maxim(n, x)
04.     r ← max – min + 1
05.     y ← zero(r)
06.     For i ← 1, n execute
07.         y[x[i] – min + 1]  ← y[x[i] – min + 1] + 1
08.     EndFor
09.     idx ← 1
10.     For i ← 1, r execute
11.         While y[i] > 0 execute
12.             x[idx] ← i + min – 1
13.             idx ← idx + 1
14.             y[i] ← y[i] - 1
15.         EndWhile
16.     EndFor
17. EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Ha az ***x*** vektor negatív számokat is tartalmaz, az algoritmus nemlétező pozíciók hozzáférésével fog próbálkozni az ***y*** vektorban.

B. Ha a 9. és a 10. sor tartalmát helyettesítenénk az alábbi utasításokkal, a `ceFace(n, x)` algoritmus ugyanazt az eredményt térítené vissza, mint az eredeti algoritmus,.

```
x[1] ← min
idx ← 2
y[1] ← y[1] – 1
For i ← 2, r execute
```

C. Ha az algoritmust `ceFace(2, [5, 8])` alakban hívjuk meg, az ***x*** vektor tartalma ***x*** = `[6, 9]` lesz.

D. A `ceFace(n, x)` algoritmus végrehajtása után, az ***x*** vektor elemei az eredeti vektor elemeinek egy permutációját fogják ábrázolni.

21. Adott a `p(x, n, a, b, c, d)` algoritmus, ahol ***x*** egy ***n*** (0 ≤ ***n*** ≤ 100) elemű egész számokat tartalmazó vektor (***x***[1], ***x***[2], ..., ***x***[***n***], ahol -100 ≤ ***x***[***i***] ≤ 100, ha ***i*** = 1, 2, ..., ***n***), valamint az ***a***, ***b***, ***c*** és ***d*** egész számok (0 ≤ ***a***, ***b***, ***c***, ***d*** ≤ 100).

```
Algorithm p(x, n, a, b, c, d):
    If n = 0 then
        Return a = b AND c = d
    EndIf
    p1 ← p(x, n - 1, a + x[n], b, c * x[n], d)
    p2 ← p(x, n - 1, a,  b + x[n], c, d * x[n])
    Return p1 OR p2
EndAlgorithm
```

Tudva azt, hogy ***x*** = `[2, 9, 5, 6, 8, 4, 1, 2, 5, 3, 4, 1, 9, 6, 8, 3]`, a következő állítások közül melyek igazak?

A. Ha az algoritmust `p(x, 16, 0, 0, 1, 1)` alakban hívjuk meg, a visszatérített érték *True*.

B. Ha az algoritmust `p(x, 16, 0, 0, 1, 1)` alakban hívjuk meg, a visszatérített érték *False*.

C. Ha az algoritmust `p(x, 16, 0, 0, 1, 1)` alakban hívjuk meg, az végtelen ciklusba kerül.

D. Ha az algoritmust `p(x, 16, 0, 0, 1, 1)` alakban hívjuk meg, az algoritmus ugyanazt az eredményt téríti vissza az ***x*** vektor bármilyen permutációjára.

22. Adottak a `rec(n, x, i, j)` és `ceFace(n, x)` algoritmusok, ahol ***n*** egy természetes szám ($1 \le \boldsymbol{n} \le 10^3$) és ***x*** egy ***n*** elemű egész számokat tartalmazó vektor (***x***[1], ***x***[2], ..., ***x***[***n***], -100 ≤ ***x***[***k***] ≤ 100, ahol ***k*** = 1, 2, ..., ***n***), valamint ***i*** és ***j*** egész számok a [0, ***n***] intervallumból. A `maxim(a, b)` algoritmus a nagyobb értéket téríti vissza ***a*** és ***b*** közül.

```
Algorithm rec(n, x, i, j):
    If i = n then
        Return 0
    EndIf
    a ← rec(n, x, i + 1, j)
    b ← 0
    If j = 0 then
        b ← 1 + rec(n, x, i + 1, i)
    Else
        If x[i] > x[j] then
            b ← 1 + rec(n, x, i + 1, i)
        EndIf
    EndIf
    Return maxim(a, b)
EndAlgorithm
```

```
Algorithm ceFace(n, x):
    Return rec(n, x, 1, 0)
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Egy szigorúan növekvő ***x*** vektorra a `ceFace(n, x)` algoritmus ***n***-t térít vissza.

B. Az algoritmus időbonyolultsága a legrosszabb esetben $O(\boldsymbol{n}^2)$.

C. Ha az algoritmust `ceFace(8, [10, 15, 9, 30, 21, 50, 42, 60])` alakban hívjuk meg, a visszatérített érték 5.

D. Ha az algoritmust `ceFace(2, [3, 2])` alakban hívjuk meg, a visszatérített érték 1.

23. Egy ***n*** számot *különlegesnek* nevezünk, ha a prímtényezői között csak a 2, 3 és 5 szerepel. Például különleges számok az 1 ($1 = 2^0 * 3^0 * 5^0$), a 12 ($12 = 2^2 * 3$) vagy a 30 ($30 = 2 * 3 * 5$). A `zero(k)` algoritmus egy ***k*** elemű vektort térít vissza, ahol minden elem egyenlő nullával.
Az A, B, C, D válaszokban található utasítássorozatok közül, melyeket kell beszúrni a `special(n)` algoritmusba a pontozott részre, úgy, hogy az algoritmus visszatérítse az ***n***. különleges számot, ahol ***n*** természetes szám ($1 \le n \le 10^5$)?

```
Algorithm special(n):
    v ← zero(n)
    v[1] ← 1; c2 ← 1; c3 ← 1; c5 ← 1
    nr ← 1
    While nr < n execute
        val1 ← v[c2] * 2
        val2 ← v[c3] * 3
        val3 ← v[c5] * 5
        If val1 ≤ val2 AND val1 ≤ val3 then
            elem ← val1
            c2 ← c2 + 1
        Else
            If val2 ≤ val1 AND val2 ≤ val3 then
                elem ← val2; c3 ← c3 + 1
            Else
                elem ← val3
                c5 ← c5 + 1
            EndIf
        EndIf
        ..........
    EndWhile
    Return v[n]
EndAlgorithm
```

A.
```
v[nr] ← elem
nr ← nr + 1
```

B.
```
If v[nr] < elem then
    v[nr + 1] ← elem
    nr ← nr + 1
EndIf
```

C.
```
nr ← nr + 1
v[nr] ← elem
```

D.
```
tmp ← nr
While elem < v[tmp] AND tmp ≥ 1 execute
    v[tmp + 1] ← v[tmp]
    tmp ← tmp - 1
EndWhile
v[tmp + 1] ← elem
nr ← nr + 1
```

24. Adott egy ***n*** természetes szám ($0 \le n \le 2^{31}$) és meg szeretnénk határozni azon bitek számát az ***n*** szám 2-es számrendszerbeli ábrázolásában pontosan 32 bitet használva, melyeknek értéke egyenlő ***k***-val ($k \in \{0, 1\}$). Az algoritmusokban a következő bitműveleteket használjuk: **&** (**AND**), **<<** (balra tolás) és **>>** (jobbra tolás) az alábbi jelentéssel:

- Ha ***x*** és ***y*** természetes számok, akkor ***x*** & ***y*** az **AND** műveletet hajtja végre minden bitre a bináris ábrázolásukon: az eredmény egy bitje csak akkor 1 ha a megfelelő bitek ***x***-ből és ***y***-ból is 1-gyel egyenlőek; egyébként 0.
- Ha ***x*** természetes szám, a ***x*** << ***i*** művelet ekvivalens ***x*** 2-vel való szorzásával ***i***-szer, és a ***x*** >> ***i*** művelet ***x*** 2-vel való osztásával ***i***-szer.

Az alábbi algoritmusok közül, melyik téríti vissza a kért értéket?

A.
```
Algorithm countBits_A(n, k):
    count ← 0
    For i ← 0, 31 execute
        If ((n & (1 << i)) >> i) = k then
            count ← count + 1
        EndIf
    EndFor
    Return count
EndAlgorithm
```

B.
```
Algorithm countBits_B(n, k):
    count ← 0
    While n > 0 execute
        If (n & 1) = 1 then
            count ← count + 1
        EndIf
        n ← n >> 1
    EndWhile
    If k = 0 then
        count ← 32 – count
    EndIf
    Return count
EndAlgorithm
```

C.
```
Algorithm countBits_C(n, k):
    If n = 0 then
        If k = 0 then
            Return 32
        Else
            Return 0
        EndIf
    Else
        If (n & 1) = k then
            Return 1 + countBits_C(n >> 1, k)
        Else
            Return countBits_C(n >> 1, k)
        EndIf
    EndIf
EndAlgorithm
```

D.
```
Algorithm countBits(n, k, poz):
    If poz < 0 then
        Return 0
    Else
        If ((n & (1 << poz)) >> poz) = k then
            Return 1 + countBits(n, k, poz - 1)
        Else
            Return countBits(n, k, poz - 1)
        EndIf
    EndIf
EndAlgorithm
Algorithm countBits_D(n, k):
    Return countBits(n, k, 31)
EndAlgorithm
```

BABEŞ-BOLYAI TUDOMÁNYEGYETEM

MATEMATIKA ÉS INFORMATIKA KAR

MATEK-INFO verseny – 2025 április 12

Informatika írásbeli

JAVÍTÁSI KULCS & MEGOLDÁSOK

HIVATALBÓL: 10 pont

1	B	3.75 pont
2	D	3.75 pont
3	ABD	3.75 pont
4	BD	3.75 pont
5	BD	3.75 pont
6	BC	3.75 pont
7	AD	3.75 pont
8	A	3.75 pont
9	BD	3.75 pont
10	AB	3.75 pont
11	BC	3.75 pont
12	AB	3.75 pont
13	D	3.75 pont
14	AD	3.75 pont
15	AC	3.75 pont
16	AC	3.75 pont
17	A	3.75 pont
18	AC	3.75 pont
19	BC	3.75 pont
20	D	3.75 pont
21	AD	3.75 pont
22	D	3.75 pont
23	B	3.75 pont
24	ABD	3.75 pont