BABEŞ-BOLYAI TUDOMÁNYEGYETEM
MATEMATIKA-INFORMATIKA KAR

Felvételi verseny – 2025 július 17

Informatika írásbeli vizsga

FONTOS MEGJEGYZÉS:
Más pontosítások hiányában:

- Minden aritmetikai műveletet végtelen adattípusokon végzünk (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 vagy karakterlánc tömbszakaszát a vektor vagy karakterlánc olyan elemei alkotják, amelyek egymás utáni pozíciókon találhatók.
- Ha ugyanabban a sorban több egymásutáni értékadó utasítás található, ezek "; "-vel vannak elválasztva.

1. Adott a `ceFace(n, x)` algoritmus, ahol n egy természetes szám ($1 \leq n \leq 10^4$), x egy n elemű, egész számokat tároló vektor ($x[1], x[2], ..., x[n]$, $-100 \leq x[i] \leq 100$, $i = 1, 2, ..., n$).

A `ceFace(n, x)` algoritmus mely implementálásai térítik vissza az x vektor legnagyobb elemét?

A.
```
Algorithm aux(n, x, i, b):
    If i > n then
        Return b
    EndIf
    If b < x[i] then
        b ← x[i]
    EndIf
    Return aux(n, x, i + 1, b)
EndAlgorithm

Algorithm ceFace(n, x):
    Return aux(n, x, 1, x[1])
EndAlgorithm
```

B.
```
Algorithm ceFace(n, x):
    i ← 1
    While i < n execute
        If x[i] ≥ x[i + 1] then
            Return False
        EndIf
        i ← i + 1
    EndWhile
    Return True
EndAlgorithm
```

C.
```
Algorithm ceFace(n, x):
    i ← 1
    b ← x[i]
    While i < n execute
        If b < x[i + 1] then
            b ← x[i + 1]
        EndIf
        i ← i + 1
    EndWhile
    Return b
EndAlgorithm
```

D.
```
Algorithm ceFace(n, x):
    i ← 1
    b ← x[i]
    While i < n execute
        If b > x[i + 1] then
            b ← x[i + 1]
        EndIf
        i ← i + 1
    EndWhile
    Return b
EndAlgorithm
```

2. Adott az `f1(a, b)` és az `f2(a, b)` algoritmus, ahol a és b természetes számok ($1 \leq a, b \leq 10^4$).

```
Algorithm f1(a, b):
    While b ≠ 0 execute
        temp ← b
        b ← a MOD b
        a ← temp
    EndWhile
    Return a
EndAlgorithm

Algorithm f2(a, b):
    Return f1(a, b) = 1
EndAlgorithm
```

Mely esetekben térít vissza *True*-t az `f2(a, b)` algoritmus?

A. Akkor és csakis akkor, ha az a és b számok a *Fibonacci* sorozat elemei.
B. Akkor és csakis akkor, ha az $a + b$ szám számjegyeinek összege egyenlő az $a - b$ szám számjegyeinek összegével
C. Akkor és csakis akkor, ha az $a - b$ szám páros számjegyeinek darabszáma egyenlő az $a + b$ szám páratlan számjegyeinek darabszámával
D. Akkor és csakis akkor, ha az a és b számok relatív prímek.

3. Adott az `f(n)` algoritmus, ahol ***n*** egy természetes szám ($3 \leq n \leq 10^3$).

```
Algorithm f(n):
    If n = 1 then
        Return 0
    Else
        Return (2 * n - 3) * (2 * n - 1) + f(n - 1)
    EndIf
EndAlgorithm
```

Az adott összegek közül melyiket számítja ki az `f(n)` algoritmus?

A. $\sum_{k=1}^{n}(2k-3)\,(2k-1)$

B. $\sum_{k=1}^{n-1}(2k-1)\,(2k+1)$

C. $\sum_{k=2}^{n-1}(2k-1)\,(2k+1)$

D. $\sum_{k=2}^{n}(2k-3)\,(2k-1)$

4. Adott az `f(x, y)` algoritmus, ahol ***x*** és ***y*** természetes számok ($3 \leq x, y \leq 10^3$):

```
Algorithm f(x, y):
    If x = 1 then
        Return y
    EndIf
    If x MOD 2 = 0 then
        If x > 0 then
            Return f(x DIV 2, y * 2)
        Else
            Return 0
        EndIf
    EndIf
    Return y + f(x DIV 2, y * 2)
EndAlgorithm
```

A következő algoritmusok közül melyik téríti vissza ugyanazt az értéket, mint az `f(x, y)` algoritmus ***x*** és ***y*** bármely értékére?

A.

```
Algorithm a(x, y):
    If x = 0 then
        Return 0
    EndIf
    If x MOD 2 = 0 then
        Return 2 * a(x DIV 2, y)
    EndIf
    Return y + a(x - 1, y)
EndAlgorithm
```

B.

```
Algorithm b(x, y):
    If x = 0 then
        Return y
    EndIf
    If x MOD 2 = 0 then
        Return b(x DIV 2, y)
    EndIf
    Return y + b(x - 1, y)
EndAlgorithm
```

C.

```
Algorithm c(x, y):
    If x = 0 then
        Return 0
    EndIf
    Return y + c(x - 1, y)
EndAlgorithm
```

D.

```
Algorithm d(x, y):
    s ← 0
    While x > 0 execute
        s ← s + y
        x ← x - 1
    EndWhile
    Return s
EndAlgorithm
```

5. Adott a `ceFace(x, n)` algoritmus, ahol ***n*** egy természetes szám ($1 \leq n \leq 10^4$), ***x*** egy ***n*** elemű, egész számokat tároló vektor ($x[1], x[2], ..., x[n]$, $-100 \leq x[i] \leq 100$, $i = 1, 2, ..., n$):

```
Algorithm ceFace(x, n):
    For i ← 1, 10 execute
        For j ← 1, n - 1 execute
            If x[j] > x[j + 1] then
                tmp ← x[j]
                x[j] ← x[j + 1]
                x[j + 1] ← tmp
            EndIf
        EndFor
    EndFor
EndAlgorithm
```

A következő állítások közül melyek lesznek igazak a `ceFace(x, n)` algoritmus meghívása után?

A. Az ***x*** vektor első pozícióját a vektor legkisebb értékű eleme foglalja el.

B. Az ***x*** vektor utolsó pozícióját a vektor legnagyobb értékű eleme foglalja el.

C. Az ***x*** vektor növekvően rendezett.

D. Ha $n > 10$, akkor az ***x*** vektor első 10 eleme növekvően rendezett.

6. Adott az n természetes szám ($1 \leq n \leq 10^4$) és az x, n elemű, egész számokat tároló vektor ($x[1], x[2], ..., x[n]$, $1 \leq x[i] \leq 10^3$, $i = 1, 2, ..., n$). A `gcd(a, b)` algoritmus az a és b természetes szám legnagyobb közös osztóját téríti vissza.

A `gcdVector(x, n)` algoritmus mely implementálásai térítik vissza az x vektor n elemének legnagyobb közös osztóját?

A.
```
Algorithm gcdVector(x, n):
    If n = 1 then
        Return 1
    EndIf
    Return gcd(x[n], gcdVector(x, n - 1))
EndAlgorithm
```

B.
```
Algorithm gcdVector(x, n):
    result ← x[1]
    For i ← 2, n execute
        result ← gcd(x[i - 1], x[i])
    EndFor
    Return result
EndAlgorithm
```

C.
```
Algorithm gcdVector(x, n):
    result ← gcd(x[1], x[n])
    i ← 2; j ← n - 1
    While i ≤ j execute
        result ← gcd(result, gcd(x[i], x[j]))
        i ← i + 1
        j ← j - 1
    EndWhile
    Return result
EndAlgorithm
```

D.
```
Algorithm gcdVector2(x, n, i):
    If i = n then
        Return x[n]
    EndIf
    Return gcd(x[i], gcdVector2(x, n, i + 1))
EndAlgorithm

Algorithm gcdVector(x, n):
    Return gcdVector2(x, n, 1)
EndAlgorithm
```

7. Adott az `afla(n, x)` algoritmus, ahol n egy természetes szám ($1 \leq n \leq 10^4$), x egy n elemű, egész számokat tároló vektor ($x[1], x[2], ..., x[n]$, $-100 \leq x[i] \leq 100$, $i = 1, 2, ..., n$):

```
Algorithm afla(n, x):
    M ← x[1]
    For i ← 1, n execute
        For j ← i, n execute
            For k ← j, n execute
                If M < x[i] + x[j] + x[k] then
                    M ← x[i] + x[j] + x[k]
                EndIf
            EndFor
        EndFor
    EndFor
    Return M
EndAlgorithm
```

Mi az algoritmus időbonyolultsága?

A. $O(3 * n)$
B. $O(n^3)$
C. $O(n / 3)$
D. $O(\log_3 n)$

8. Adott a `ceFace(n)` algoritmus, ahol n egy természetes szám ($1 \leq n \leq 100$). A `min(a, b)` algoritmus az a és b számok közül a kisebbet téríti vissza.

```
Algorithm ceFace(n):
    c1 ← 0; c2 ← 0
    For i ← 1, n execute
        v ← i
        While v MOD 2 = 0 execute
            c1 ← c1 + 1
            v ← v DIV 2
        EndWhile
        While v MOD 5 = 0 execute
            c2 ← c2 + 1
            v ← v DIV 5
        EndWhile
    EndFor
    Return min(c1, c2)
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Az algoritmus az [1, n] intervallumban található páros számok darabszámát téríti vissza.
B. Az algoritmus az [1, n] intervallumban található 5-tel osztható számok darabszámát téríti vissza.
C. Az algoritmus az első n természetes szám összegének végén található egymásutáni 0-ák darabszámát téríti vissza.
D. A fenti válaszok közül egyik sem helyes.

9. Adott az n ($2 \leq n \leq 10^5$) elemű, egész számokat tároló x vektor ($x[1], x[2], ..., x[n]$, $-100 \leq x[i] \leq 100$, $i = 1, 2, ..., n$). A `max(a, b)` algoritmus az a és b számok közül a nagyobbat téríti vissza.

A `maxPePozitiePara(x, n)` algoritmus mely implementálásai térítik vissza azt a legnagyobb elemet, amely páros értékű pozíción található a vektorban?

A.
```
Algorithm maxPePozitiePara(x, n):
    maxVal ← x[2]
    For i ← 4, n, 2 execute
        If x[i] > maxVal then
            maxVal ← x[i]
        EndIf
    EndFor
    Return maxVal
EndAlgorithm
```

B.
```
Algorithm maxPePozitiePara(x, n):
    maxVal ← -100
    For i ← 1, n, 2 execute
        If x[i] > maxVal then
            maxVal ← x[i]
        EndIf
    EndFor
    Return maxVal
EndAlgorithm
```

C.
```
Algorithm maxPePozitiePara2(x, n, i):
    If i > n then
        Return -100
    EndIf
    maxRest ← maxPePozitiePara2(x, n, i + 2)
    Return max(x[i], maxRest)
EndAlgorithm

Algorithm maxPePozitiePara(x, n):
    Return maxPePozitiePara2(x, n, 1)
EndAlgorithm
```

D.
```
Algorithm maxPePozitiePara2(x, n, i):
    If i > n then
        Return -100
    EndIf
    maxRest ← maxPePozitiePara2(x, n, i + 2)
    Return max(x[i], maxRest)
EndAlgorithm

Algorithm maxPePozitiePara(x, n):
    Return maxPePozitiePara2(x, n, 2)
EndAlgorithm
```

10. Legyenek a következő igaznak tekintett feltételezések:

1. Anna csak akkor megy sétálni, ha süt a nap.
2. Mária csak akkor megy sétálni, ha Anna is megy.
3. Ha süt a nap, Tudor megy sétálni.

Az alábbi következtetések közül melyeket lehet igaznak tekinteni a feltételezések alapján?

A. Ha Mária megy sétálni, akkor Tudor is megy.
B. Ha nem süt a nap, akkor Anna nem megy sétálni.
C. Ha süt a nap, akkor Mária megy sétálni.
D. Ha nem süt a nap, akkor Tudor nem megy sétálni.

11. Adott kettes számrendszerben az $x = 10000110111_{(2)}$ szám, és 4-es számrendszerben az $y = 11011_{(4)}$ szám.

Mi lesz az $x + y$ szám értéke 10-es számrendszerben?

A. 1079 B. 1404 C. 2285 D. Egyik sem az A, B és C változatok közül

12. Adott a `ceFace(n, a)` algoritmus, ahol n egy természetes szám ($1 \leq n \leq 10^4$), a egy n elemű, természetes számokat tároló vektor ($a[1], a[2], ..., a[n]$, $1 \leq a[i] \leq 10^9$, $i = 1, 2, ..., n$).

```
 1. Algorithm ceFace(n, a):
 2.     m ← 0
 3.     For i ← 1, n execute
 4.         nr ← 1
 5.         While a[i] > 9 execute
 6.             nr ← nr * 10
 7.             a[i] ← a[i] DIV 10
 8.         EndWhile
 9.         If m < a[i] * nr then
10.             m ← a[i] * nr
11.         EndIf
12.     EndFor
13.     Return m
14. EndAlgorithm
```

A következő állítások közül melyek igazak?

A. A `ceFace(5, [222, 2043, 29, 2, 20035])` hívás eredményeként az algoritmus 2000-et térít vissza.
B. Függetlenül n és a értékétől, a `ceFace(n, a)` algoritmus egy olyan számot térít vissza, amely nem található meg az a vektorban.
C. A `ceFace(5, [34, 254, 21, 543, 123])` hívás eredményeként az algoritmus 500-at térít vissza.
D. Abból, ha a 10. sorban található utasítás n-szer hajtódik végre, következik, hogy az eredeti a vektor növekvően rendezett volt.

13. Létrehozunk egy irányítatlan gráfot a következőképpen: minden n természetes szám esetén ($2 \leq n \leq 20$) hozzáadunk a gráfhoz egy n-nel címkézett csomópontot, valamint két x és y címkéjű csomópontot összekötünk egy éllel, ha y valódi osztója x-nek.

A következő állítások közül melyek igazak?

A. A létrehozott gráf 5 összefüggő komponensből áll.
B. Csak a 12-es, 18-as és 20-as címkéjű csomópontok foka maximális.
C. Bármely 2 x és y szám esetében, amelyek nem prímszámok és $2 \leq x \leq y \leq n$, az x foka nagyobb vagy egyenlő y fokával.
D. Egyetlen olyan csomópont létezik, amelynek foka 1.

14. Adott a `ceva(n, v)` algoritmus, ahol v egy n ($10 \leq n \leq 10^5$) elemű, egész számokat tároló vektor ($v[1]$, $v[2]$, ..., $v[n]$, $-10 \leq v[i] \leq 10$, $i = 1, 2, \ldots n$). A `zero(k)` algoritmus egy k elemű vektort téríti vissza, amelynek minden eleme 0.

```
 1. Algorithm ceva(n, v):
 2.     fr ← zero(21)
 3.     s ← 0
 4.     For i ← 1, n execute
 5.         If fr[v[i] + 11] = 0 then
 6.             s ← s + 1
 7.         EndIf
 8.         fr[v[i] + 11] ← 1
 9.     EndFor
10.     Write s
11.     p ← 1
12.     For i ← 1, n execute
13.         If fr[v[i] + 11] = 1 then
14.             p ← p * v[i]
15.             fr[v[i] + 11] ← 0
16.         EndIf
17.     EndFor
18.     Return p
19. EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Ha a 10. sorban kiírt érték nagyobb, mint 10, a visszatérített eredmény egy páros szám.
B. Ha a 10. sorban kiírt érték nagyobb, mint 11, a visszatérített eredmény egy negatív szám.
C. Ha a 10. sorban kiírt érték 21, a visszatérített eredmény egyenlő $(10!)^2$ értékével.
D. A lehetséges legnagyobb érték, amely kiírható a 10. sorban, amelynek esetében a visszatérített szorzat nem végződik 0-val, a 16.

15. Adott az `f(x)` algoritmus, ahol x egy természetes szám ($0 \leq x \leq 10^5$).

```
Algorithm f(x):
    If x ≤ 1 then
        Return 1
    EndIf
    Return f(x - 1) + f(x - 2)
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Az `f(10)` hívás eredményeképpen az `f(x)` algoritmus 177-szer hívja meg önmagát, beszámítva az eredeti hívást is.
B. Az `f(x)` hívás eredményeképpen a visszatérített érték a Fibonacci sorozat (1, 1, 2, 3, 5, 8, 13, …) eleme.
C. Az `f(10)` hívás eredményeképpen a visszatérített érték 89.
D. Az `f(5)` hívás következtében összesen 4 összeadás hajtódik végre.

16. Adott a `ceva(A, n, r, c, nr, x)` algoritmus, ahol n egy természetes szám ($1 < n \leq 10$), A egy $n \times n$ méretű mátrix, amelynek elemei természetes számok ($A[1][1]$, $A[1][2]$, ..., $A[n][n]$), r, c, x természetes számok ($1 \leq r, c, x \leq 10$) és nr egy egész szám ($-10^3 \leq nr \leq 10^3$). Ha $n = 4$ és eredetileg $A[3][2] = 5$ és $A[1][4] = 8$, az alábbi hívássorozatok közül melyik módosítja az A mátrix elemeit úgy, hogy $A[3][2]$ legyen 50 és $A[1][4]$ legyen 16?

```
Algorithm ceva(A, n, r, c, nr, x):
    If (r + x - 1 ≤ n) AND (c + x - 1 ≤ n) then
        For i ← r, r + x - 1 execute
            For j ← c, c + x - 1 execute
                A[i][j] ← A[i][j] * nr
            EndFor
        EndFor
    EndIf
EndAlgorithm
```

A.
```
ceva(A, n, 3, 2, 5, 1)
ceva(A, n, 2, 1, 4, 3)
ceva(A, n, 3, 3, 16, 2)
```

B.
```
ceva(A, n, 1, 3, 2, 2)
ceva(A, n, 3, 2, 2, 3)
ceva(A, n, 2, 1, 10, 3)
```

C.
```
ceva(A, n, 2, 3, 4, 2)
ceva(A, n, 1, 1, 2, 4)
ceva(A, n, 3, 1, 2, 1)
ceva(A, n, 2, 1, 5, 3)
```

D. Egyik hívássorozat sem vezet a kijelentésben megadott értékekhez.

17. Adott a `cauta(x, n, e)` algoritmus, ahol ***x*** egy ***n*** elemű ($1 \leq n \leq 10^4$), természetes számokat tároló vektor (***x***[1], ***x***[2], ..., ***x***[***n***], ***i*** = 1, 2, ..., ***n***, $0 \leq x[i] \leq 10^4$) és ***e*** egy természetes szám ($1 \leq e \leq 10^4$).

A következő algoritmusok közül melyik térít vissza *True*-t akkor és csakis akkor, ha az ***e*** szám megtalálható az ***x*** vektorban?

A.
```
Algorithm cauta(x, n, e):
    g ← False; i ← 1
    While i ≤ n execute
        g ← (x[i] = e)
        i ← i + 1
    EndWhile
    Return g
EndAlgorithm
```

B.
```
Algorithm cauta(x, n, e):
    g ← False; i ← 1
    While NOT g AND i ≤ n execute
        g ← (x[i] = e)
        i ← i + 1
    EndWhile
    Return g
EndAlgorithm
```

C.
```
Algorithm cauta(x, n, e):
    c ← 0
    For i ← 1, n execute
        If x[i] = e then
            c ← c + 1
        Else
            c ← c - 1
        EndIf
    EndFor
    Return n ≠ -c
EndAlgorithm
```

D.
```
Algorithm cauta(x, n, e):
    g ← False; i ← 1
    While i ≤ n execute
        If x[i] < e + 1 AND x[i] MOD e = 0 then
            g ← True
        EndIf
        i ← i + 1
    EndWhile
    Return g
EndAlgorithm
```

18. Adott az ***m*** és ***n*** ($0 \leq m, n \leq 10$) természetes szám és az `Ack(m, n)` algoritmus, amely az Ackerman-függvény értékét számítja ki az ***m*** és ***n*** paraméterekre.

```
Algorithm Ack(m, n):
    If m = 0 then
        Return n + 1
    EndIf
    If m > 0 AND n = 0 then
        Return Ack(m - 1, 1)
    EndIf
    If m > 0 AND n > 0 then
        Return Ack(m - 1, Ack(m, n - 1))
    EndIf
EndAlgorithm
```

Hányszor hívja meg önmagát az `Ack(1, 4)` algoritmus?

A. 9-szer hívja meg önmagát.
B. Ugyanannyiszor, mint az `Ack(1, 2)` hívás esetében.
C. 4-gyel többször, mint az `Ack(1, 2)` hívás esetében.
D. 11-szer hívja meg önmagát.

19. Adott a `P(s, n)` algoritmus, ahol ***s*** egy ***n*** hosszú karakterlánc ($3 < n \leq 100$). A `copy(s, poz, cate)` algoritmus az ***s*** karakterláncnak azt a tömbszakaszát téríti vissza, amely ***cate*** karaktert tartalmaz a ***poz*** pozíciótól kezdődően, ahol $0 \leq cate \leq n$, és $1 \leq poz \leq n - cate + 1$. A karakterláncok esetében a "+" operátor a konkatenálást (összefűzést) jelenti.

```
Algorithm P(s, n):
    i ← n DIV 2
    j ← n DIV i
    If n MOD 2 = 0 then
        s ← copy(s, i, i - 1) +
                    copy(s, j, j - 1)
    Else
        s ← copy(s, i, i - 2) +
                  copy(s, j, j - 2) +
                       copy(s, 1, 1)
    EndIf
    Return s
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Ha ***s*1** és ***s*2** két karakterlánc, amelyeknek a hossza ***n***, illetve ***m***, $n \neq m$, a `P(s1, n)` és a `P(s2, m)` hívások két, különböző hosszúságú karakterláncot térítenek vissza.
B. Ha az ***n*** hosszú ***s*** karakterlánc csak `'a'`, `'b'` és `'c'` karaktereket tartalmaz, 252 különböző értéke létezik ***s***-nek, amelyekre a `P(s, n)` hívás az `"ac"` karakterláncot téríti vissza.
C. A `P("concurs de admitere", 19)` hívás eredményeképpen az algoritmus a `"de admic"` karakterláncot téríti vissza.
D. Ha az ***n*** hosszú ***s*** karakterlánc csak `'0'` és `'1'` karaktereket tartalmaz, 72 különböző értéke létezik ***s***-nek, amelyekre a `P(s, n)` hívás az `"101"` karakterláncot téríti vissza.

20. Besatírozzuk az ***A*** és ***B*** téglalapok által lefedett területeket. A téglalapokat a bal alsó sarkuk és a jobb felső sarkuk által azonosítjuk be. Az ***A*** téglalap esetén a koordináták (***ax1***, ***ay1***) és (***ax2***, ***ay2***), a ***B*** téglalap esetén a koordináták pedig (***bx1***, ***by1***) és (***bx2***, ***by2***), $0 \leq ax1, ay1, ax2, ay2, bx1, by1, bx2, by2 \leq 10^4$, $ax1 < ax2$, $ay1 < ay2$, $bx1 < bx2$, $by1 < by2$. A `min(a, b)` és `max(a, b)` algoritmusok az ***a*** és ***b*** számok közül a kisebbet, illetve a nagyobbat térítik vissza.

Az alábbi algoritmusok közül melyik számítja ki a besatírozott rész területét?

A.
```
Algorithm calculArie(ax1, ay1, ax2, ay2, bx1, by1,
                                              bx2, by2):
    a1 ← (ax2 - ax1) * (ay2 - ay1)
    a2 ← (bx2 - bx1) * (by2 - by1)
    xSup ← max(0, min(ax2, bx2) - max(ax1, bx1))
    ySup ← max(0, min(ay2, by2) - max(ay1, by1))
    aSup ← xSup * ySup
    arie ← a1 + a2 - aSup
    Return arie
EndAlgorithm
```

B.
```
Algorithm calculArie(ax1, ay1, ax2, ay2, bx1, by1,
                                              bx2, by2):
    a1 ← (ax2 - ax1) * (ay2 - ay1)
    a2 ← (bx2 - bx1) * (by2 - by1)
    If ax2 ≤ bx1 OR bx2 ≤ ax1 OR ay2 ≤ by1 OR
                                        by2 ≤ ay1 then
        aSup ← 0
    Else
        xSup ← min(ax2, bx2) - max(ax1, bx1)
        ySup ← min(ay2, by2) - max(ay1, by1)
        aSup ← xSup * ySup
    EndIf
    arie ← a1 + a2 - aSup
    Return arie
EndAlgorithm
```

C.
```
Algorithm calculArie(ax1, ay1, ax2, ay2, bx1, by1,
                                              bx2, by2):
    arie ← (ax2 - ax1) * (ay2 - ay1)
    If ax2 ≤ bx1 OR bx2 ≤ ax1 OR ay2 ≤ by1
                                    OR by2 ≤ ay1 then
        arie ← arie + (bx2 - bx1) * (by2 - by1)
    EndIf
    Return arie
EndAlgorithm
```

D.
```
Algorithm calculArie(ax1, ay1, ax2, ay2, bx1, by1,
                                              bx2, by2):
    a1 ← (ax2 - ax1) * (ay2 - ay1)
    a2 ← (bx2 - bx1) * (by2 - by1)
    aSup ← (min(ax2, bx2) - max(ax1, bx1)) +
                    (min(ay2, by2) - max(ay1, by1))
    arie ← a1 + a2 - aSup
    Return arie
EndAlgorithm
```

21. Adott az `f(A, B, m, n, k)` algoritmus, ahol ***m***, ***n*** és ***k*** természetes számok ($1 \leq m, n, k \leq 100$), ***A*** egy ***n*** elemű, egész számokat tároló vektor (***A***[1], ***A***[2], ..., ***A***[***n***], ahol $-100 \leq A[i] \leq 100$, $i = 1, 2, ..., n$), és ***B*** egy ***n*** elemű vektor, ahol minden elem egyenlő nullával.

```
Algorithm f(A, B, m, n, k):
    aux ← 0
    For j ← 1, k - 1 execute
        aux ← aux + B[j] * A[j]
    EndFor
    If aux = m then
        Write "Solutie: "
        For j ← 1, k - 1 execute
            If B[j] = 1 then
                Write A[j], " "
            EndIf
        EndFor
        Write new line
    Else
        If k ≠ n + 1 then
            For i ← 0, 1 execute
                B[k] ← i
                f(A, B, m, n, k + 1)
            EndFor
        EndIf
    EndIf
EndAlgorithm
```

Ha ***A*** = [4, 5, 8, 9, 3, 12, 15, 7], az `f(A, B, 12, 8, 1)` hívás következtében az első kiírt eredmény `Solutie: 12.`

Melyik lesz a második és a harmadik kiírt eredmény?

A.
```
Solutie: 4 8
Solutie: 4 5 3
```

B.
```
Solutie: 4 8
Solutie: 5 7
```

C.
```
Solutie: 9 3
Solutie: 5 7
```

D.
```
Solutie: 9 3
Solutie: 4 8
```

22. Adott az `oareCe(n1)` algoritmus, ahol ***n*1** egy természetes szám ($0 \leq \boldsymbol{n1} \leq 10^6$).

```
Algorithm oareCe(n1):
    n2 ← 0
    While n1 > n2 execute
        n3 ← n1 MOD 10
        n2 ← n2 * 10 + n3
        n1 ← n1 DIV 10
    EndWhile
    If n1 = n2 then
        Return True // (*)
    EndIf
    Return n1 = (n2 DIV 10)
EndAlgorithm
```

A következő állítások közül melyek igazak?

A. Az `oareCe(1210)` hívás eredményeként az algoritmus *False*-t térít vissza.
B. Az `oareCe(282)` hívás a (*)-gal jelölt sor végrehajtását eredményezi.
C. A [101, 500] intervallumban 4 különböző ***n*** természetes szám létezik, amelyekre az `oareCe(n)` a (*)-gal jelölt sor végrehajtódik.
D. Ha az algoritmust az `oareCe(((i * 6) DIV 7) * ((j * 7) DIV (i + j)))` alakban hívjuk meg, ahol ***i*** = 1 és ***j*** = 2, *True*-t térít vissza.

23. Adott az `interesant(a, b, c, n)` algoritmus, ahol ***c*** és ***n*** természetes számok ($1 \leq \boldsymbol{n} \leq 100$, $1 \leq \boldsymbol{c} \leq 10^4$), ***a*** és ***b*** két ***n*** elemű, egész számokat tároló vektor (***a***[1], ***a***[2], ..., ***a***[***n***] és ***b***[1], ***b***[2], ..., ***b***[***n***], $1 \leq \boldsymbol{a}[\boldsymbol{i}], \boldsymbol{b}[\boldsymbol{i}] \leq 100$, ***i*** = 1, 2, ..., ***n***). A `max(x, y)` algoritmus az ***x*** és ***y*** számok közül a nagyobbat téríti vissza.

```
Algorithm interesant(a, b, c, n):
    If n = 0 OR c = 0 then
        Return 0
    EndIf
    If a[n] > c then
        Return interesant(a, b, c, n - 1)
    EndIf
    x ← b[n] + interesant(a, b, c - a[n], n - 1)
    y ← interesant(a, b, c, n - 1)
    Return max(x, y)
EndAlgorithm
```

Mely értéket téríti vissza az `interesant([2, 3, 1], [6, 10, 3], 5, 3)` hívás eredményeképpen az algoritmus?

A. 13
B. 10
C. 16
D. 19

24. Adott a `divide(x, y)` algoritmus, amelynek ***x*** és ***y*** egész osztási hányadosát kellene kiszámítania, ahol ***x*** és ***y*** egész számok ($-10^5 \leq \boldsymbol{x}, \boldsymbol{y} \leq 10^5, \boldsymbol{y} \neq 0$). Az `a << n` művelet az ***a*** szám bitjeit balra tolja ***n*** pozícióval, így a művelet ekvivalens a szám 2^n-nel történő szorzásával.

```
Algorithm divide(x, y):
    negativ ← 1
    If x < 0 then
        negativ ← -negativ
        x ← -x
    EndIf
    If y < 0 then
        negativ ← -negativ
        y ← -y
    EndIf
    cat ← 0
    While x ≥ y execute
        temp ← y
        multiplu ← 1
        While x ≥ (temp << 1) execute
            temp ← temp << 1
            ... // (1)
        EndWhile
        x ← x - temp
        ... // (2)
    EndWhile
    ... // (3)
    Return cat
EndAlgorithm
```

Mely utasításokat kell beszúrnunk az (1), (2) és (3)-mal jelzett sorokba, ahhoz, hogy a `divide(x, y)` algoritmus a helyes eredményt térítse vissza?

A.
```
(1): multiplu ← multiplu << 1
(2): cat ← cat + multiplu
(3): If negativ = -1 then
         cat ← -cat
     EndIf
```

B.
```
(1): multiplu ← multiplu << 1
(2): cat ← cat + 1
(3): cat ← -negativ * cat
```

C.
```
(1): multiplu ← multiplu * 2
(2): cat ← cat + 1
(3): cat ← -negativ * cat
```

D.
```
(1): multiplu ← multiplu * 2
(2): cat ← cat + multiplu
(3): cat ← negativ * cat
```

BABEŞ-BOLYAI TUDOMÁNYEGYETEM

MATEMATIKA ÉS INFORMATIKA KAR

Felvételi verseny – 2025 július 17

Informatika írásbeli

JAVÍTÁSI KULCS & MEGOLDÁSOK

HIVATALBÓL: 10 pont

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