**UNIVERSITATEA BABEŞ-BOLYAI**
**FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ**

# Concurs de admitere – 17 iulie 2025

**Proba scrisă la Informatică**

NOTĂ IMPORTANTĂ:
În lipsa altor precizări:

- Toate operațiile aritmetice se efectuează pe tipuri de date nelimitate (nu există *overflow* / *underflow*).
- Numerotarea indicilor tuturor vectorilor, matricelor și a șirurilor de caractere începe de la 1.
- Toate restricțiile se referă la valorile parametrilor actuali la momentul apelului inițial.
- O subsecvență a unui vector sau a unui șir de caractere este formată din elemente care ocupă poziții consecutive în vector, respectiv în șirul de caractere.
- Dacă pe un același rând apar mai multe instrucțiuni de atribuire consecutive, acestea sunt delimitate prin "; ".

**1.** Se consideră algoritmul `ceFace(n, x)`, unde $n$ este număr natural ($1 \le n \le 10^4$), iar $x$ este un vector cu $n$ elemente numere întregi ($x[1], x[2], ..., x[n]$, $-100 \le x[i] \le 100$, pentru $i = 1, 2, ..., n$).

Care dintre următoarele implementări ale algoritmului `ceFace(n, x)` returnează cea mai mare valoare din vectorul $x$?

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.** Se consideră algoritmii `f1(a, b)` și `f2(a, b)`, unde $a$ și $b$ sunt numere naturale ($1 \le a, b \le 10^4$).

```
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
```

În ce situații returnează *True* algoritmul `f2(a, b)`?

A. Dacă și numai dacă numerele $a$ și $b$ sunt numere din șirul lui *Fibonacci*.
B. Dacă și numai dacă suma cifrelor numărului $a + b$ este egal cu suma cifrelor numărului $a - b$.
C. Dacă și numai dacă numărul cifrelor pare ale numărului $a - b$ este egal cu numărul cifrelor impare ale numărului $a + b$.
D. Dacă și numai dacă numerele $a$ și $b$ sunt două numere relativ prime.

**3.** Se consideră algoritmul `f(n)`, unde $n$ este un număr natural ($3 \leq n \leq 10^3$).

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

Care dintre sumele date sunt calculate de algoritmul `f(n)`?

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.** Se consideră algoritmul `f(x, y)`, unde $x$ și $y$ sunt numere naturale ($3 \leq x, y \leq 10^3$):

```
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
```

Care dintre următorii algoritmi returnează aceeași valoare ca algoritmul `f(x, y)` pentru orice valori ale lui $x$ și $y$?

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.** Se consideră algoritmul `ceFace(x, n)`, unde $n$ este număr natural ($1 \leq n \leq 10^4$), iar $x$ este un vector cu $n$ elemente numere întregi ($x[1], x[2], ..., x[n]$, $-100 \leq x[i] \leq 100$, pentru $i = 1, 2, ..., n$):

```
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
```

Care dintre următoarele afirmații vor fi adevărate după apelul `ceFace(x, n)`?

A. Prima poziție a vectorului $x$ este ocupată de elementul minim al vectorului.

B. Ultima poziție a vectorului $x$ este ocupată de elementul maxim al vectorului.

C. Vectorul $x$ este sortat crescător.

D. Dacă $n > 10$, atunci primele 10 elemente ale vectorului $x$ sunt ordonate crescător.

**6.** Se consideră numărul natural $n$ ($1 \leq n \leq 10^4$) și vectorul $x$ cu $n$ elemente numere naturale ($x[1], x[2], ..., x[n]$, $1 \leq x[i] \leq 10^3$, pentru $i = 1, 2, ..., n$). Algoritmul `gcd(a, b)` returnează cel mai mare divizor comun al numerelor naturale $a$ și $b$.

Care din următoarele implementări ale algoritmului `gcdVector(x, n)` returnează cel mai mare divizor comun al celor $n$ elemente din vectorul $x$?

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.** Se consideră algoritmul `afla(n, x)`, unde $n$ este număr natural ($1 \leq n \leq 10^4$), iar $x$ este un vector cu $n$ elemente numere întregi ($x[1], x[2], ..., x[n]$, $-100 \leq x[i] \leq 100$, pentru $i = 1, 2, ..., n$):

```
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
```

Care este complexitatea timp a algoritmului?

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

**8.** Se consideră algoritmul `ceFace(n)`, unde $n$ este număr natural ($1 \leq n \leq 100$). Algoritmul `min(a, b)` returnează valoarea minimă dintre numerele $a$ și $b$.

```
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
```

Care dintre următoarele afirmații sunt adevărate?

A. Algoritmul returnează numărul numerelor pare din intervalul $[1, n]$.
B. Algoritmul returnează numărul numerelor divizibile cu 5 din intervalul $[1, n]$.
C. Algoritmul returnează numărul de cifre 0 consecutive aflate la finalul sumei primelor $n$ numere naturale.
D. Niciunul dintre răspunsurile de mai sus nu este corect.

**9.** Se dă vectorul ***x*** cu ***n*** elemente ($2 \le n \le 10^5$) numere întregi (***x***[1], ***x***[2], ..., ***x***[***n***], $-100 \le x[i] \le 100$, pentru $i = 1, 2, ..., n$). Algoritmul `max(a, b)` returnează valoarea maximă dintre numerele ***a*** și ***b***.

Care dintre implementările următoare ale algoritmului `maxPePozitiePara(x, n)` returnează cel mai mare element aflat pe o poziție pară în vector?

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.** Considerăm următoarele ipoteze ca fiind adevărate:

1. Ana merge la plimbare doar dacă e soare.
2. Maria merge la plimbare doar dacă merge și Ana.
3. Dacă e soare, Tudor merge la plimbare.

Care dintre următoarele concluzii se deduc din ipoteze?

A. Dacă Maria merge la plimbare, atunci și Tudor merge.
B. Dacă nu e soare, atunci Ana nu merge la plimbare.
C. Dacă e soare, atunci Maria merge la plimbare.
D. Dacă nu e soare, atunci Tudor nu merge la plimbare.

**11.** Se consideră numărul $x = 10000110111_{(2)}$ dat în baza 2 și numărul $y = 11011_{(4)}$ dat în baza 4.

Care este valoarea sumei $x + y$ în baza 10?

A. 1079 B. 1404 C. 2285 D. Niciuna din variantele A, B și C

**12.** Se consideră algoritmul `ceFace(n, a)`, unde ***n*** este un număr natural ($1 \le n \le 10^4$), iar ***a*** este un vector cu ***n*** elemente numere naturale (***a***[1], ***a***[2], ..., ***a***[***n***], $1 \le a[i] \le 10^9$, pentru $i = 1, 2, ..., n$).

```
 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
```

Care dintre următoarele afirmații sunt adevărate?

A. În urma apelului `ceFace(5, [222, 2043, 29, 2, 20035])`, algoritmul returnează valoarea 2000.
B. Indiferent de valorile lui ***n*** și ***a***, algoritmul `ceFace(n, a)` returnează un număr care nu face parte din vectorul ***a***.
C. În urma apelului `ceFace(5, [34, 254, 21, 543, 123])`, algoritmul returnează valoarea 500.
D. Dacă instrucțiunea de pe rândul 10 se execută de ***n*** ori, rezultă că vectorul ***a***, dat inițial, era sortat strict crescător.

**13.** Construim un graf neorientat în modul următor: pentru fiecare număr natural $n$, astfel încât $2 \leq n \leq 20$, adăugăm un nod etichetat cu acel număr, iar între două noduri cu etichete $x$ și $y$ adăugăm o muchie, dacă $y$ este divizor propriu al lui $x$.

Care dintre următoarele afirmații sunt adevărate?

A. Graful construit are 5 componente conexe.
B. Doar nodurile 12, 18 și 20 au gradul maxim.
C. Pentru oricare 2 numere neprime $x$, $y$ ($2 \leq x \leq y \leq n$), gradul lui $x$ este mai mare sau egal decât gradul lui $y$.
D. Există un singur nod cu gradul 1.

**14.** Se consideră algoritmul `ceva(n, v)`, unde $v$ este un vector de $n$ ($10 \leq n \leq 10^5$) numere întregi ($v[1]$, $v[2]$, ..., $v[n]$, $-10 \leq v[i] \leq 10$, pentru $i = 1, 2, \ldots n$). Algoritmul `zero(k)` returnează un vector cu $k$ elemente, toate egale cu zero.

```
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
```

Care din următoarele afirmații sunt adevărate?

A. Dacă valoarea afișată pe linia 10 este mai mare ca 10, rezultatul returnat este un număr par.
B. Dacă valoarea afișată pe linia 10 este mai mare ca 11, rezultatul returnat este un număr negativ.
C. Dacă valoarea afișată pe linia 10 este 21, rezultatul returnat este egal cu $(10!)^2$.
D. Cea mai mare valoare posibilă afișată pe linia 10, pentru care produsul returnat nu se termină cu cifra 0, este 16.

**15.** Se consideră algoritmul `f(x)`, unde $x$ este număr natural ($0 \leq x \leq 10^5$).

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

Care dintre următoarele afirmații sunt adevărate?

A. În urma apelului `f(10)` se vor efectua în total 177 de apeluri ale algoritmului `f(x)`, incluzând apelul inițial.
B. Valoarea returnată în urma apelului `f(x)` face parte din șirul Fibonacci (1, 1, 2, 3, 5, 8, 13, …).
C. Valoarea returnată în urma apelului `f(10)` este 89.
D. În urma apelului `f(5)` se vor efectua în total 4 adunări.

**16.** Se consideră algoritmul `ceva(A, n, r, c, nr, x)`, unde $n$ este număr natural ($1 < n \leq 10$), $A$ este o matrice cu $n \times n$ elemente numere naturale ($A[1][1]$, $A[1][2]$, ..., $A[n][n]$), $r$, $c$, $x$ sunt numere naturale ($1 \leq r, c, x \leq 10$), iar $nr$ este număr întreg ($-10^3 \leq nr \leq 10^3$). Dacă $n = 4$ și inițial $A[3][2] = 5$ și $A[1][4] = 8$, care din secvențele de apeluri de mai jos va modifica elementele matricei $A$ astfel încât $A[3][2]$ să fie egal cu 50 și $A[1][4]$ să fie egal cu 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.
Niciuna din secvențele de apeluri nu produce modificarea precizată în enunț.

**17.** Se consideră algoritmul `cauta(x, n, e)`, unde ***x*** este un vector cu ***n*** ($1 \le n \le 10^4$) elemente numere naturale (***x***[1], ***x***[2], ..., ***x***[***n***], pentru ***i*** = 1, 2, ..., ***n***, $0 \le x[i] \le 10^4$) și ***e*** este un număr natural ($1 \le e \le 10^4$).

Care dintre următorii algoritmi returnează *True* dacă și numai dacă numărul ***e*** se găsește în vectorul ***x***?

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.** Se consideră numerele naturale ***m*** și ***n*** ($0 \le m, n \le 10$) și algoritmul `Ack(m, n)` care calculează valoarea funcției Ackerman pentru parametrii ***m*** și ***n***.

```
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
```

Câte autoapeluri se vor efectua în urma apelului `Ack(1, 4)`?

A. Se vor efectua 9 autoapeluri.
B. Același număr de autoapeluri ca în urma apelului `Ack(1, 2)`
C. Cu 4 autoapeluri mai mult decât în urma apelului `Ack(1, 2)`
D. Se vor efectua 11 autoapeluri.

**19.** Se consideră algoritmul `P(s, n)`, unde ***s*** este un șir de ***n*** caractere ($3 < n \le 100$). Algoritmul `copy(s, poz, cate)` returnează o subsecvență a șirului ***s***, formată din ***cate*** caractere, începând de la poziția ***poz***, unde $0 \le cate \le n$, respectiv $1 \le poz \le n - cate + 1$. Pentru șiruri de caractere, operatorul "+" reprezintă operația de concatenare.

```
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
```

Care dintre următoarele afirmații sunt adevărate?

A. Dacă ***s*1** și ***s*2** sunt două șiruri de caractere de lungime ***n*** respectiv ***m***, $n \ne m$, în urma apelurilor `P(s1, n)` și `P(s2, m)`, algoritmul va returna două șiruri de caractere de lungimi diferite.
B. Dacă șirul de caractere ***s*** de lungime ***n*** conține doar caracterele `'a'`, `'b'` și `'c'`, există 252 de valori distincte pentru ***s***, pentru care, în urma apelului `P(s, n)`, algoritmul va returna șirul `"ac"`.
C. În urma apelului `P("concurs de admitere", 19)`, algoritmul returnează valoarea `"de admic"`.
D. Dacă șirul de caractere ***s*** de lungime ***n*** conține doar caractere `'0'` și `'1'`, există 72 de valori distincte pentru ***s***, pentru care, în urma apelului `P(s, n)`, algoritmul va returna șirul `"101"`.

**20.** Hașurăm ariile acoperite de două dreptunghiuri ***A*** și ***B.*** Fiecare dreptunghi este definit de colțul său din stânga jos și colțul din dreapta sus. Pentru dreptunghiul ***A***, coordonatele sunt (***ax1***, ***ay1***) și (***ax2***, ***ay2***), iar pentru dreptunghiul ***B***, coordonatele sunt (***bx1***, ***by1***) și (***bx2***, ***by2***), $0 \leq ax1, ay1, ax2, ay2, bx1, by1, bx2, by2 \leq 10^4$, $ax1 < ax2$, $ay1 < ay2$, $bx1 < bx2$, $by1 < by2$. Algoritmii `min(a, b)` și `max(a, b)` returnează valoarea minimă, respectiv maximă, dintre două numere ***a*** și ***b***.

Care dintre algoritmii de mai jos calculează aria suprafeței hașurate?

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.** Se consideră algoritmul `f(A, B, m, n, k)`, unde ***m***, ***n*** și ***k*** sunt numere naturale ($1 \leq m, n, k \leq 100$), iar ***A*** este un vector cu ***n*** elemente numere întregi ($A[1], A[2], ..., A[n]$, unde $-100 \leq A[i] \leq 100$, pentru $i = 1, 2, ..., n$), iar ***B*** este un vector cu ***n*** elemente, toate egale cu zero.

```
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
```

Dacă ***A*** = [4, 5, 8, 9, 3, 12, 15, 7], în urma apelului `f(A, B, 12, 8, 1)` prima soluție afișată va fi `Solutie: 12.`

Care va fi a doua și a treia soluție afișată?

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.** Se consideră algoritmul `oareCe(n1)`, unde ***n*1** este număr natural ($0 \leq n1 \leq 10^6$).

```
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
```

Care din următoarele afirmații sunt adevărate?

A. În urma apelului `oareCe(1210)`, algoritmul va returna valoarea *False*.
B. Apelul `oareCe(282)` duce la executarea liniei marcate (*)
C. Există 4 valori distincte ale lui ***n***, număr natural din intervalul [101, 500] pentru care apelul `oareCe(n)` duce la executarea liniei marcate (*).
D. Dacă algoritmul se apelează sub forma `oareCe(((i * 6) DIV 7) * ((j * 7) DIV (i + j)))`, unde ***i*** = 1 și ***j*** = 2, algoritmul returnează *True*.

**23.** Se consideră algoritmul `interesant(a, b, c, n)`, unde ***c*** și ***n*** sunt numere naturale ($1 \leq n \leq 100$, $1 \leq c \leq 10^4$), iar ***a*** și ***b*** sunt doi vectori de lungime ***n***, cu elemente numere întregi (***a***[1], ***a***[2], ..., ***a***[***n***] și ***b***[1], ***b***[2], ..., ***b***[***n***], $1 \leq a[i], b[i] \leq 100$, pentru $i = 1, 2, ..., n$). Algoritmul `max(x, y)` returnează valoarea maximă dintre numerele ***x*** și ***y***.

```
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
```

Ce valoare se returnează în urma apelului `interesant([2, 3, 1], [6, 10, 3], 5, 3)`?

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

**24.** Se consideră algoritmul `divide(x, y)` care calculează câtul împărțirii întregi a numărului ***x*** la ***y***, unde ***x*** și ***y*** sunt numere întregi ($-10^5 \leq x, y \leq 10^5, y \neq 0$). Operația `a << n` deplasează biții numărului ***a*** spre stânga cu ***n*** poziții, care este echivalent cu înmulțirea numărului cu $2^n$.

```
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
```

Ce instrucțiuni trebuie inserate pe liniile marcate cu (1), (2) și (3) pentru ca algoritmul `divide(x, y)` să returneze rezultatul corect?

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
```

UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

*Concurs de admitere – 17 iulie 2025*

*Proba scrisă la INFORMATICĂ*

# BAREM ȘI REZOLVARE

**OFICIU**: 10 puncte

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