UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Concurs de admitere – 19 iulie 2021

Proba scrisă la Informatică

1. Fie următorul subalgoritm, având ca parametru de intrare numărul natural n și care returnează un număr natural.

```
Subalgoritm calcul(n):
    E ← 1
    P ← 1
    i ← 2
    CâtTimp i ≤ n execută
        P ← (-1) * P * i
        E ← E + P
        i ← i + 1
    SfCâtTimp
    returnează E
SfSubalgoritm
```

Care este valoarea returnată de subalgoritm, în condițiile în care $n \geq 1$?

A. $1! - 2! + 3! - 4! + ... + (-1)^{n+1} \cdot n!$
B. $1 - 1! + 2! - 3! + ... + (-1)^{n} \cdot n!$
C. $1 - 1 \cdot 2 + 1 \cdot 2 \cdot 3 - 1 \cdot 2 \cdot 3 \cdot 4 + ... + (-1)^{n+1} \cdot 1 \cdot 2 \cdot ... \cdot n$
D. $1 + 1 \cdot 2 - 1 \cdot 2 \cdot 3 + 1 \cdot 2 \cdot 3 \cdot 4 + ... + (-1)^{n} \cdot 1 \cdot 2 \cdot ... \cdot n$

2. Un fișier Excel conține n înregistrări numerotate de la 1 la n. Aceste înregistrări trebuie copiate într-un fișier Word în care înregistrările se vor aranja în câte r rânduri și c coloane pe fiecare pagină (cu excepția primei și ultimei pagini). Pe prima pagină a documentului Word, datorită prezenței unui antet, numărul de rânduri este r_1, $r_1 < r$ (numărul de rânduri prezent pe prima pagina este mai mic).

Înregistrările vor fi aranjate în fișierul Word pe fiecare pagină de sus în jos pe fiecare coloană, coloanele fiind completate de la stânga la dreapta: dacă prima înregistrare de pe o pagină are numărul de ordine i, înregistrarea cu numărul de ordine (i + 1) va fi prezentă sub ea, iar înregistrarea cu numărul de ordine (**i** + **r**) va fi prima înregistrare de pe coloana 2 de pe pagina respectivă ș.a.m.d.

Pentru n = 5000, r = 46, r_1 = 12 și c = 2 pe ce pagină a documentului Word și pe ce coloană se va regăsi înregistrarea cu număr de ordine i = 3245?

A. Pagina 36, ultima coloană
B. Pagina 37, prima coloană
C. Pagina 37, ultima coloană
D. Pagina 38, prima coloană

3. Se consideră subalgoritmul `ceFace(m)`, unde m este un număr natural ($10 \leq m \leq 10000$).

```
Subalgoritm ceFace(m):
    Dacă m = 0 atunci
        returnează 0
    SfDacă
    Dacă m MOD 9 = 0 atunci
        returnează 9
    SfDacă
    returnează m MOD 9
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Subalgoritmul returnează restul împărțirii numărului ***m*** la 9.
B. Subalgoritmul returnează numărul divizorilor care sunt divizibili cu 9 ai numărului ***m***.
C. Subalgoritmul returnează cifra de control a numărului ***m*** (suma cifrelor sale, apoi suma cifrelor acestei sume, până când suma obținută este un număr format dintr-o singură cifră).
D. Subalgoritmul returnează cifra de control a numărului ***m*** (suma cifrelor sale, apoi suma cifrelor acestei sume, până când suma obținută este un număr format dintr-o singură cifră) dacă și numai dacă numărul ***m*** este divizibil cu 9.

4. Pentru a genera numerele cu ***n*** cifre formate doar din cifrele 0, 2, 9, se utilizează un algoritm care, pentru ***n*** = 2, generează în ordine crescătoare numerele 20, 22, 29, 90, 92, 99.
Dacă ***n*** = 4 și se utilizează același algoritm, care este numărul generat imediat după numărul 2009?

A. 2022
B. 2090
C. 2010
D. Niciuna dintre celelalte variante

5. Se consideră subalgoritmul `cauta(n)`, unde ***n*** este un număr natural ($0 \leq n \leq 1000000$).

```
Subalgoritm cauta(n):
    v ← 0
    Dacă n = 0 atunci
        returnează 1
    altfel
        m ← n
        CâtTimp m > 0 execută
            Dacă m MOD 10 = 0 atunci
                v ← v + 1
            SfDacă
            m ← m DIV 10
        SfCâtTimp
        returnează v
   SfDacă
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Subalgoritmul determină și returnează câte cifre are numărul ***n***.
B. Subalgoritmul returnează 1 dacă numărul ***n*** este o putere a lui 10 și 0 altfel.
C. Subalgoritmul returnează 1 dacă numărul ***n*** se termină cu cifra 0 și 0 altfel.
D. Subalgoritmul determină și returnează numărul de cifre 0 din numărul ***n***.

6. Se consideră subalgoritmul `abc(a, n, p)`, unde ***n*** este număr natural ($1 \leq n \leq 10000$), ***p*** este număr întreg ($-10000 \leq p \leq 10000$), iar ***a*** este un șir cu ***n*** numere naturale nenule (***a***[1], ***a***[2], …, ***a***[n]).

```
Subalgoritm abc(a, n, p):
    Dacă n < 1 atunci
        returnează 0
    altfel
        Dacă (1 ≤ p) ȘI (p ≤ n) atunci
            returnează a[p]
        altfel
            returnează -1
        SfDacă
    SfDacă
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Subalgoritmul returnează -1 dacă și numai dacă ***p*** este negativ sau mai mare decât ***n***.
B. Subalgoritmul returnează elementul de pe poziția ***p*** dacă ***p*** este strict mai mare decât 0 și mai mic sau egal decât lungimea șirului.
C. Subalgoritmul nu returnează niciodată 0 pentru valori ale parametrilor care respectă precondițiile din enunț.
D. Subalgoritmul returnează elementul de pe poziția ***p*** dacă ***p*** este mai mare sau egal cu 0 și mai mic strict decât lungimea șirului. În cazul în care ***p*** nu este între 1 și ***n***, returnează -1.

7. Care dintre secvențele următoare determină în variabila ***i*** lungimea unui șir de caractere care se termină cu caracterul '*' (asterisc)? Primul caracter se află la indicele 1, iar caracterul asterisc este parte a șirului de caractere.

A.
```
i ← 1
CâtTimp x[i] ≠ '*' execută
    i ← i + 1
SfCâtTimp
```
B.
```
i ← 1
CâtTimp x[i] = '*' execută
    i ← i + 1
SfCâtTimp
i ← i - 1
```
C.
```
i ← 1
CâtTimp x[i] ≠ '*' execută
    i ← i + 1
SfCâtTimp
i ← i + 1
```
D.
```
i ← 1
CâtTimp x[i] ≠ '*' execută
    i ← i + 1
SfCâtTimp
i ← i - 1
```

8. Fie următorul subalgoritm, având ca parametru numărul natural nenul ***n*** și care returnează un număr natural.

```
Subalgoritm f(n):
    j ← n
    CâtTimp j > 1 execută
        i ← 1
        CâtTimp i ≤ n execută
            i ← 2 * i
        SfCâtTimp
        j ← j DIV 3
    SfCâtTimp
    returnează j
SfSubalgoritm
```

În care dintre următoarele clase de complexitate se încadrează complexitatea timp a algoritmului?

A. $O(\log_2 \boldsymbol{n})$
B. $O(\log_2^2 \boldsymbol{n})$
C. $O(\log_3^2 \boldsymbol{n})$
D. $O(\log_2 \log_3 \boldsymbol{n})$

9. Subalgoritmul `cate(n, m)` primește ca parametri numerele naturale ***n*** și ***m***.

```
Subalgoritm cate(n, m):
    Dacă n ≤ m atunci
        Dacă (n MOD 2 = 0) ȘI (n MOD 3 ≠ 0) atunci
             returnează 1 + cate(n + 1, m)
        altfel
             returnează cate(n + 1, m)
        SfDacă
    altfel
       returnează 0
    SfDacă
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Dacă ***n*** = 0 și ***m*** = 1, subalgoritmul returnează valoarea 0.
B. Dacă ***n*** = 4 și ***m*** = 21, subalgoritmul returnează valoarea 6.
C. Dacă ***n*** = 7 și ***m*** = 120, subalgoritmul returnează valoarea 36.
D. Dacă ***n*** = 1 și ***m*** = 215, subalgoritmul returnează valoarea 72.

10. Se consideră subalgoritmul `verifica(n)`, unde ***n*** este un număr natural (1 ≤ ***n*** ≤ 100000).

```
Subalgoritm verifica(n):
  CâtTimp n > 0 execută
    Dacă (n MOD 3) > 1 atunci
      returnează 0
    SfDacă
    n ← n DIV 3
  SfCâtTimp
  returnează 1
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Subalgoritmul returnează 1 dacă ***n*** este o putere a lui 3, 0 în caz contrar.
B. Subalgoritmul returnează 1 dacă scrierea în baza 3 a lui ***n*** conține doar cifrele 0 și/sau 1, 0 în caz contrar.
C. Subalgoritmul returnează 1 dacă ***n*** poate fi scris ca sumă a puterilor distincte ale lui 3, 0 în caz contrar.
D. Subalgoritmul returnează 1 dacă scrierea în baza 3 a lui ***n*** conține doar cifra 2, 0 în caz contrar.

11. Pentru un număr natural ***nr*** (1000 ≤ ***nr*** ≤ 1000000), definim operația de decrementare în modul următor: dacă ultima cifră a lui ***nr*** nu este 0, scădem 1 din ***nr***, altfel, împărțim ***nr*** la 10 și păstrăm doar partea întreagă. Care dintre următorii subalgoritmi returnează, la apelul `decrementare(nr, k)`, numărul obținut aplicând de ***k*** ori (0 ≤ ***k*** ≤ 100) operația de decrementare pe numărul ***nr***? De exemplu, pentru ***nr*** = 15243 și ***k*** = 10, rezultatul este 151.

A.
```
Subalgoritm decrementare(nr, k):
    Dacă k = 0 atunci
        returnează nr
    altfel
        Dacă nr MOD 10 ≠ 0 atunci
            returnează decrementare(nr DIV 10, k - 1)
        altfel
            returnează decrementare(nr - 1, k - 1)
        SfDacă
    SfDacă
SfSubalgoritm
```

B.
```
Subalgoritm decrementare(nr, k):
    CâtTimp k > 0 execută
        Dacă nr MOD 10 = 0 atunci
            nr ← nr DIV 10
        altfel
            nr ← nr - 1
        SfDacă
    SfCâtTimp
    returnează nr
SfSubalgoritm
```

C.
```
Subalgoritm decrementare(nr, k):
    Pentru i ← 1, k execută
        Dacă nr MOD 10 > 0 atunci
            nr ← nr - 1
        altfel
            nr ← nr DIV 10
        SfDacă
    SfPentru
    returnează nr
SfSubalgoritm
```

D.
```
Subalgoritm decrementare(nr, k):
    Dacă k = 0 atunci
        returnează nr
    altfel
        Dacă k > nr MOD 10 atunci
            nr1 ← nr DIV 10
            returnează decrementare(nr1,  k - nr MOD 10 - 1)
        altfel
            returnează decrementare(nr - k, 0)
        SfDacă
    SfDacă
SfSubalgoritm
```

12. Se dă următorul subalgoritm care are ca parametri de intrare un șir ***x*** cu ***n*** numere naturale (***x***[1], ***x***[2], …, ***x***[***n***]) și numărul întreg ***n***.

```
Subalgoritm f(x, n):
    Dacă n = 1 atunci
        returnează 100
    altfel
        Dacă x[n] > f(x, n - 1) atunci
            returnează x[n]
        altfel
            returnează f(x, n - 1)
        SfDacă
    SfDacă
SfSubalgoritm
```

Care va fi rezultatul execuției subalgoritmului pentru ***x*** = [101, 7, 6, 3] și ***n*** = 4?

A. 101
B. 3
C. 100
D. 7

13. Subalgoritmul de mai jos are ca parametri de intrare un șir ***a*** cu ***n*** numere naturale (***a***[1], ***a***[2], …, ***a***[***n***]) și numărul natural ***n*** ($2 \leq n \leq 10000$).

```
Subalgoritm h(a, n):
    Dacă n ≤ 0 atunci
        returnează 0
    SfDacă
    Dacă (n MOD 2 = 0) ȘI (a[n] MOD 2 = 0) atunci
        returnează h(a, n - 1) + a[n]
    SfDacă
    returnează h(a, n - 1) - a[n]
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Subalgoritmul returnează diferența dintre suma elementelor care au aceeași paritate cu poziția pe care se află și suma elementelor care au paritate diferită față de poziția pe care se află din șirul ***a***.
B. Subalgoritmul returnează diferența dintre suma elementelor pare de pe pozițiile pare și suma elementelor impare de pe pozițiile impare din șirul ***a***.
C. Subalgoritmul returnează diferența dintre suma elementelor pare și suma elementelor impare din șirul ***a***.
D. Subalgoritmul returnează diferența dintre suma elementelor pare de pe poziții pare și suma celorlalte elemente din șirul ***a***.

14. Se consideră subalgoritmul `ceFace(n)`, cu parametrul ***n*** număr natural nenul.

```
Subalgoritm ceFace(n):
    i ← 1
    CâtTimp n > 0 execută
        Dacă n MOD 2 ≠ 0 atunci
            scrie i
        SfDacă
        i ← i + 1
        n ← n DIV 2
    SfCâtTimp
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Subalgoritmul afișează secvența: 12345 pentru ***n*** = 31.
B. Subalgoritmul afișează secvența: 234 pentru ***n*** = 14.
C. Subalgoritmul afișează 1 la începutul secvenței, pentru ***n*** impar.
D. Subalgoritmul afișează un singur număr pentru $n = 2^k$, unde ***k*** este un număr natural.

15. Se dă o mulțime ***S***, care conține ***n*** intervale specificate prin capătul stâng s_i și capătul drept d_i ($s_i < d_i\ \forall\ i = 1...n$). Se dorește determinarea unei submulțimi $S' \subseteq S$ de ***m*** elemente, astfel încât să nu existe două intervale în ***S'*** care se intersectează și ***m*** să aibă cea mai mare valoare posibilă.
Care dintre următoarele strategii rezolvă corect problema?

A. Se sortează intervalele din mulțimea ***S*** crescător după capătul stâng. Se adaugă primul interval din șirul sortat în ***S'***. Se parcurg celelalte elemente din șir în ordinea sortată și când se întâlnește un interval care nu se intersectează cu intervalul care a fost adăugat ultima oară în ***S'***, se adaugă și acesta în ***S'***.
B. Se sortează intervalele din mulțimea ***S*** crescător după capătul drept. Se adaugă primul interval din șirul sortat în ***S'***. Se parcurg celelalte elemente din șir în ordinea sortată și când se întâlnește un interval care nu se intersectează cu intervalul care a fost adăugat ultima oară în ***S'***, se adaugă și acesta în ***S'***.

C. Se sortează intervalele din mulțimea ***S*** crescător după lungimea intervalului. Se adaugă primul interval din șirul sortat în ***S'***. Se parcurg celelalte elemente din șir în ordinea sortată și când se întâlnește un interval care nu se intersectează cu intervalul care a fost adăugat ultima oară în ***S'***, se adaugă și acesta în ***S'***.

D. Se sortează intervalele din mulțimea ***S*** crescător după numărul de intervale din ***S*** cu care se intersectează. Se adaugă primul interval din șirul sortat în ***S'***. Se parcurg celelalte elemente din șir în ordinea sortată și când se întâlnește un interval care nu se intersectează cu intervalul care a fost adăugat ultima oară în ***S'***, se adaugă și acesta în ***S'***.

16. Se consideră subalgoritmul `f(a, b)`, care primește ca parametri două numere naturale ***a*** și ***b*** ($1 \leq$ ***a*** $<$ ***b*** ≤ 1000).

```
Subalgoritm f(a, b):
    m ← 0
    Pentru n ← a, b execută
        c ← 0
        Pentru d ← 1, n execută
            Dacă n MOD d = 0 atunci
                c ← c + 1
            SfDacă
        SfPentru
        Dacă c > m atunci
            m ← c
        SfDacă
    SfPentru

    Pentru n ← a, b execută
        c ← 0
        Pentru d ← 1, n execută
            Dacă n MOD d = 0 atunci
                c ← c + 1
            SfDacă
        SfPentru
        Dacă c = m atunci
            scrie n
        SfDacă
    SfPentru
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Subalgoritmul afișează maximul dintre numărul de divizori ai lui ***a*** și numărul de divizori ai lui ***b***.
B. Subalgoritmul afișează numerele naturale din intervalul [***a***, ***b***] care au proprietatea că au cel mai mare număr de divizori.
C. Subalgoritmul afișează numărul de divizori pentru fiecare număr natural din intervalul [***a***, ***b***].
D. Subalgoritmul afișează numerele naturale din intervalul [***a***, ***b***] care au proprietatea că au cel mai mare număr de divizori proprii.

17. Fie numerele naturale ***a*** și ***b***, cu ***b*** $\neq 0$. Care dintre următoarele variante calculează:

- `a DIV b`, dacă `a MOD b = 0`
- `(a / b)` rotunjit în sus către cel mai apropiat întreg, dacă `a MOD b ≠ 0`

A. `(a - 1) DIV b`
B. `(a + b + 1) DIV b`
C. `(a + b - 1) DIV b`
D. `((a + 2 * b - 1) DIV b) - 1`

18. Ionel trebuie să implementeze algoritmul de căutare binară a unui element ***a*** într-un șir ***V*** cu ***n*** ($1 \leq$ ***n*** ≤ 1000) numere întregi ordonate crescător (***V***[1], ***V***[2], …, ***V***[***n***]). El scrie următorul subalgoritm:

```
Subalgoritm CautareBinara(a, n, V):
     st ← 1
     dr ← n
     Câttimp dr – st > 1 execută
          m ← (st + dr) DIV 2
          Dacă a ≤ V[m] atunci
               dr ← m
          altfel
               st ← m
          SfDacă
     SfCâttimp
     returnează dr
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Dacă ***n*** = 1 atunci valoarea returnată de subalgoritm este întotdeauna 1.
B. Pentru orice ***n*** ≥ 1, subalgoritmul scris de Ionel returnează valoarea 1 atunci când ***a*** este mai mic decât toate elementele din șir.
C. Atunci când elementul ***a*** apare în șir, subalgoritmul scris de Ionel NU returnează întotdeauna poziția (indicele în vectorul ***V***) pe care acesta apare.
D. Pentru orice ***n*** > 1, subalgoritmul scris de Ionel returnează valoarea ***n*** atunci când ***a*** este mai mare decât toate elementele din șir.

19. Se consideră subalgoritmul `calcul(x, n)`, unde parametrii de intrare sunt numerele naturale ***n*** și ***x***, cu condiția $1 \leq x \leq n < 10$.

```
Subalgoritm calcul(x, n):
    b ← 1
    Pentru i ← 1, n - x execută
        b ← b * i
    SfPentru
    a ← b
    Pentru i ← n – x + 1, n execută
        a ← a * i
    SfPentru
    returnează a DIV b
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Dacă ***n*** = 5 și ***x*** = 2, atunci subalgoritmul returnează 20.
B. Dacă ***n*** = 3 și ***x*** = 2, atunci subalgoritmul returnează 6.
C. Subalgoritmul returnează cardinalitatea mulțimii $\{\overline{c_1 c_2 \dots c_x}\ : c_i \neq c_j\ \forall\ 1 \leq i, j \leq x, i \neq j,\ 1 \leq c_i \leq n\}$
D. Subalgoritmul efectuează ***n*** operații de înmulțire.

20. Se consideră subalgoritmul `ceFace(n,k)`, care primește ca și parametru două numere naturale nenule ***n*** și ***k*** ($1 \leq n, k \leq 1000000$).

```
Subalgoritm ceFace(n, k):
    Câttimp n ≥ 1 execută
        Dacă k ≤ n atunci
            i ← k
        altfel
            i ← n
        SfDacă
```

```
                n ← n - i
                x ← 1
                Câttimp i ≥ 1 execută
                    Scrie x,' '
                    x ← x + 1
                    i ← i - 1
                SfCâttimp
            SfCâttimp
        SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Pentru ***n*** = 8 și ***k*** = 3 subalgoritmul afișează șirul 1 2 3 1 2 3 1 2
B. Pentru ***k*** = 2, cea mai mică valoare a lui ***n*** pentru care se afișează de 3 ori valoarea 1 pe ecran este ***n*** = 3.
C. Pentru ***k*** = 5, cea mai mică valoare a lui ***n*** pentru care se afișează de 37 ori valoarea 2 pe ecran este ***n*** = 182.
D. Pentru ***n*** = 7 și ***k*** = 3 subalgoritmul afișează 1 2 3 1 2 3

21. Se consideră subalgoritmul `calculeaza(a, b, c)`, cu parametrii de intrare numere naturale nenule, care calculează cel mai mare divizor comun al celor trei numere.
Care dintre următoarele sunt implementări corecte ale subalgoritmului:

A.
```
Subalgoritm calculeaza(a, b, c):
    CâtTimp (a ≠ b) SAU (a ≠ c) SAU (b ≠ c) execută
        x ← a
        Dacă a ≠ x atunci
            a ← a - x
        SfDacă
        Dacă b ≠ x atunci
            b ← b - x
        SfDacă
        Dacă c ≠ x atunci
            c ← c - x
        SfDacă
    SfCâtTimp
    returnează x
SfSubalgoritm
```

B.
```
Subalgoritm calculeaza(a, b, c):
    x ← a
    y ← b
    CâtTimp x ≠ y execută
        Dacă x > y atunci
            x ← x - y
        altfel
            y ← y - x
        SfDacă
    SfCâtTimp
    z ← c
    CâtTimp x ≠ z execută
        Dacă x > z atunci
            x ← x - z
        altfel
            z ← z - x
        SfDacă
    SfCâtTimp
    returnează x
SfSubalgoritm
```

C.
```
Subalgoritm calculeaza(a, b, c):
    CâtTimp (a ≠ b) SAU (a ≠ c) SAU (b ≠ c) execută
        x ← a
        Dacă b < x atunci
            x ← b
        SfDacă
        Dacă c < x atunci
            x ← c
        SfDacă
        Dacă a ≠ x atunci
            a ← a - x
        SfDacă
        Dacă b ≠ x atunci
            b ← b - x
        SfDacă
        Dacă c ≠ x atunci
            c ← c - x
        SfDacă
    SfCâtTimp
    returnează x
SfSubalgoritm
```

D.
```
Subalgoritm calculeaza(a, b, c):
    x ← a
    y ← b
    r ← x MOD y
    CâtTimp r ≠ 0 execută
        x ← y
        y ← r
        r ← x MOD y
    SfCâtTimp
    z ← c
    r ← y MOD z
    CâtTimp r ≠ 0 execută
        y ← z
        z ← r
        r ← y MOD z
   SfCâtTimp
   returnează z
SfSubalgoritm
```

22. Subalgoritmul `ceFace(n)` are ca parametru numărul natural $\boldsymbol{n}$ ($1 \leq \boldsymbol{n} \leq 100$).

```
Subalgoritm ceFace(n):
    s ← 0
    Dacă n MOD 2 = 0 atunci
        a ← 1
        CâtTimp a < n execută
            s ← s + a
            a ← a + 2
        SfCâtTimp
    altfel
        b ← 2
        CâtTimp b < n execută
            s ← s + b
            b ← b + 2
        SfCâtTimp
    SfDacă
    returnează s
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Dacă ***n*** este par, subalgoritmul returnează suma numerelor naturale mai mici strict decât ***n***; dacă ***n*** este impar, returnează suma numerelor naturale pare mai mici decât ***n***.
B. Dacă ***n*** este par, subalgoritmul returnează suma numerelor naturale pare mai mici strict decât ***n***; dacă ***n*** este impar, returnează suma numerelor naturale impare mai mici decât ***n***.
C. Dacă ***n*** este par, subalgoritmul returnează suma numerelor naturale impare mai mici decât ***n***; dacă ***n*** este impar, returnează suma numerelor naturale pare mai mici decât ***n***.
D. Dacă ***n*** este par, subalgoritmul returnează suma numerelor naturale pare mai mici strict decât ***n***; dacă ***n*** este impar, returnează suma numerelor naturale mai mici strict decât ***n***.

23. Subalgoritmul `ceFace(a)` primește ca parametru numărul natural ***a*** (1 ≤ ***a*** ≤ 100000).

```
Subalgoritm ceFace(a):
    b ← 0
    c ← 0
    d ← 0
    e ← 1
    CatTimp a > 0 execută
        d ← a MOD 10
        Dacă (d ≠ 4) ȘI (d < 7) atunci
            b ← b + e * (d DIV 2)
            c ← c + e * (d - d DIV 2)
        altfel
            b ← b + e
            c ← c + e * (d - 1)
        SfDacă
        a ← a DIV 10
        e ← e * 10
    SfCâtTimp
    Scrie b
    Scrie c
SfSubalgoritm
```

Care dintre următoarele perechi de valori nu vor fi afișate pentru nici o valoare de intrare validă?

A. 1112 și 11233
B. 1111 și 88888
C. 21001 și 33011
D. 3141 și 3258

24. Se consideră subalgoritmii `f(n, c)` și `g(n, c)`, care primesc ca parametri numerele naturale ***n*** și ***c***.

```
Subalgoritm f(n, c):
    Dacă n ≤ 9 atunci
        Dacă n = c atunci
            returnează 1
        altfel
            returnează 0
        SfDacă
    altfel
        Dacă n MOD 10 = c atunci
            returnează f(n DIV 10, c) + 1
        altfel
            returnează f(n DIV 10, c)
        SfDacă
    SfDacă
SfSubalgoritm
```

```
Subalgoritm g(n, c):
    Dacă c = 0 atunci
        returnează 0
    altfel
        Dacă f(n, c) > 0 atunci
            returnează g(n, c - 1) + 1
        altfel
            returnează g(n, c - 1)
        SfDacă
    SfDacă
SfSubalgoritm
```

Ce returnează apelul `g(n, 9)`?

A. Returnează numărul de cifre ale numărului ***n***.
B. Returnează numărul de cifre distincte ale numărului ***n***.
C. Returnează numărul de cifre mai mari decât 1 ale numărului ***n***.
D. Niciunul dintre celelalte răspunsuri nu este corect.

25. Pe un site fiecare utilizator înregistrat are în loc de parolă un cod secret alcătuit din ***n*** cifre. Pentru a se loga pe site, utilizatorul nu trebuie să introducă codul complet, ci pagina generează aleator 3 poziții distincte, ***p***1, ***p***2 și ***p***3, astfel încât 1 ≤ ***p***1 < ***p***2 < ***p***3 ≤ n iar utilizatorul trebuie să introducă doar cifrele de pe acele 3 poziții. De exemplu, dacă codul utilizatorului este 987654321 și pagina generează aleator pozițiile 2, 5 și 7, utilizatorul trebuie să introducă cifrele 8, 5, 3.
Mai jos aveți valorile introduse de un utilizator pentru 9 logări pe această pagină.
1, 2, 3
2, 9, 0
6, 3, 2
2, 0, 2
1, 4, 7
9, 3, 2
4, 4, 3
4, 3, 1
5, 6, 0
Presupunând că toate cele 9 logări sunt valide și codul utilizatorului nu a fost schimbat între timp, precizați care dintre următoarele afirmații sunt adevărate:

A. Codul utilizatorului sigur nu conține cifra 8.
B. Cel mai scurt cod posibil are 12 cifre.
C. Cel mai scurt cod posibil conține cifra 2 de minimum 3 ori.
D. Suma cifrelor în cel mai scurt cod posibil poate fi 44.

26. Se consideră subalgoritmul `f(x, n)` unde ***x***, ***n*** sunt numere naturale și ***x*** > 0.

```
Subalgoritm f(x, n):
    Dacă n = 0 atunci
        returnează 1
    SfDacă
    m ← n DIV 2
    p ← f(x, m)
    Dacă n MOD 2 = 0 atunci
        returnează p * p
    SfDacă
    returnează x * p * p
SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Subalgoritmul returnează x^n.
B. Dacă în loc de "`n MOD 2`" ar fi "`m MOD 2`" atunci subalgoritmul ar returna x^n.
C. Liniile după autoapelul funcției nu se vor executa niciodată.
D. Subalgoritmul returnează x^n dacă și numai dacă n este par.

27. Se consideră subalgoritmul `f2(a,b)` cu parametrii ***a*** și ***b*** numere naturale, și subalgoritmul `f(arr, i, n, p)` având ca parametri șirul ***arr*** cu ***n*** numere întregi (***arr***[1], ***arr***[2], …, ***arr***[n]), și numerele întregi ***i*** și ***p***.

```
Subalgoritm f2(a, b):
    Dacă a > b atunci
        returnează a
    altfel
        returnează b
    SfDacă
SfSubalgoritm

Subalgoritm f(arr, i, n, p):
    Dacă i = n atunci
        returnează 0
    SfDacă
    n1 ← f(arr, i + 1, n, p)
    n2 ← 0
    Dacă p + 1 ≠ i atunci
        n2 ← f(arr, i + 1, n, i) + arr[i]
    SfDacă
    returnează f2(n1, n2)
SfSubalgoritm
```

Precizați care este rezultatul apelului `f(arr, 1, 9, -10)`, dacă șirul ***arr*** conține valorile (10, 1, 3, 4, 8, 12, 1, 11, 6).

A. 24
B. 37
C. 26
D. 56

28. Fie subalgoritmul `verifica(n)`, care primește ca parametru un număr întreg ***n*** ($1 \leq n \leq 100000$) și returnează adevărat dacă ***n*** conține o cifră care este egală cu suma celorlalte cifre. De exemplu, `verifica(1517)` returnează adevărat pentru că 7 = 1 + 5 + 1.
Care din următoarele variante reprezintă implementări corecte ale subalgoritmului `verifica(n)`?

A.
```
Subalgoritm verifica(n):
    s ← 0
    c ← n
    r ← fals
    CâtTimp c > 0 execută
        s ← s + c MOD 10
        c ← c DIV 10
    SfCâtTimp
    c ← n
    CâtTimp c > 0 execută
        d ← c MOD 10
        Dacă d = s - d atunci
            r ← adevărat
        altfel
            r ← fals
        SfDaca
        c ← c DIV 10
    SfCâtTimp
    returnează r
SfSubalgoritm
```

B.
```
Subalgoritm verifica(n):
    m ← -1
    c ← n
    r ← fals
    CâtTimp c > 0 execută
        d ← c MOD 10
        c ← c DIV 10
        Dacă d > m atunci
            m ← d
        SfDacă
    SfCâtTimp
    c ← n
    s ← 0
    CâtTimp c > 0 execută
        d ← c MOD 10
        Dacă d ≠ m atunci
            s ← s + d
        SfDacă
        c ← c DIV 10
    SfCâtTimp
    Dacă s = m atunci
        r ← adevărat
    SfDacă
    returnează r
SfSubalgoritm
```

C.
```
Subalgoritm verifica(n):
    v ← [0,0,0,0,0,0,0,0,0]
    r ← fals
    CâtTimp n > 0 execută
        d ← n MOD 10
        Dacă d > 0 atunci
            v[d] ← v[d] + 1
        SfDacă
        n ← n DIV 10
    SfCâtTimp
    m ← 9
    CâtTimp v[m] = 0 execută
        m ← m - 1
    SfCâtTimp
    Dacă v[m] = 1 atunci
        d ← m
        s ← 0
        m ← m - 1
        CâtTimp m > 0 execută
            s ← s + v[m] * m
            m ← m - 1
        SfCâtTimp
        Dacă d = s atunci
            r ← adevarat
        SfDacă
    SfDacă
    returnează r
SfSubalgoritm
```

D. Niciuna dintre celelalte variante nu este corectă.

29. Se consideră subalgoritmul `f(x, n, e, y, m)`, care primește ca parametri un șir ***x*** cu ***n*** elemente numere întregi (***x***[1], ***x***[2], …, ***x***[***n***]), un șir ***y*** cu ***m*** elemente numere întregi (***y***[1], ***y***[2], …, ***y***[***m***]), și un număr întreg ***e*** care nu aparține șirului ***y*.** Subalgoritmul returnează un șir și un număr natural. Se dau subalgoritmii:

- `(c, p) ← concatenare(a, n, b, m)` care are ca parametri de intrare un șir ***a*** cu ***n*** elemente numere întregi și un șir ***b*** cu ***m*** elemente numere întregi, și returnează șirul ***c*** cu ***p*** elemente numere întregi care reprezintă concatenarea celor două șiruri ***a*** și ***b***, adică: ***a***[1], ***a***[2], …, ***a***[n], ***b***[1], ***b***[2], …, ***b***[***m***]
- `(c, p) ← diferență(a, n, b, m)` care are ca parametri de intrare un șir ***a*** cu ***n*** elemente numere întregi și un șir ***b*** cu ***m*** elemente numere întregi, și returnează șirul ***c*** cu ***p*** elemente numere întregi care conține toate elementele din șirul ***a*** (elementele rămase în șir păstrându-și ordinea inițială) care nu sunt în șirul ***b***

```
1.  Subalgoritm f(x, n, e, y, m):
2.      Dacă n = 0 atunci
3.          returnează [], 0
4.      SfDacă
5.      Dacă x[1] ≠ e atunci
6.          s ← []
7.          s[1] ← x[1]
8.          (r1, l1) ← diferență(x, n, s, 1)
9.          (r2, l2) ← f(r1, l1, e, y, m)
10.         (r3, l3) ← concatenare(s, 1, r2, l2)
11.         returnează r3, l3
12.     altfel
13.         (r1, l1) ← f(y, m, e, x, n)
14.         s ← []
15.         s[1] ← x[1]
16.         (r2, l2) ← diferență(x, n, s, 1)
17.         (r3, l3) ← f(r2, l2, e, y, m)
18.         (r4, l4) ← concatenare(r1, l1, r3, l3)
19.         returnează r4, l4
20.     SfDacă
21. SfSubalgoritm
```

Precizați care dintre următoarele afirmații sunt adevărate:

A. Subalgoritmul `f(x, n, e, y, m)` construiește un tablou unidimensional pornind de la șirul ***x*** în care aparițiile elementului **e** sunt șterse și în locul fiecărei apariții sunt inserate elementele din ***y***. Subalgoritmul returnează tabloul construit și dimensiunea acestuia.

B. Dacă șirurile ***x*** și ***y*** nu au elemente comune, atunci șirul returnat de subalgoritmul `f(x, n, e, y, m)` va conține doar elemente distincte.

C. Lungimea șirului returnat de subalgoritmul `f(x, n, e, y, m)`, având ca parametri de intrare șirurile ***x*** și ***y*** nevide, poate fi mai mică decât ***n***.

D. Dacă pe linia 18, în loc de ***r1*** și ***l1*** am avea ***y*** și ***m*** atunci funcția ar returna un tablou unidimensional (și dimensiunea lui) care ar începe cu elementele din ***y***, urmate de elementele din ***x***, aparițiile lui ***e*** fiind înlocuite de elementele din ***y***.

30. Se dă subalgoritmul `s(a, b, c)`, unde ***a***, ***b***, ***c*** sunt numere naturale nenule, ***b*** ≥ ***a***

```
Subalgoritm s(a, b, c):
    Dacă c = 0 atunci
        returnează 1
    altfel
        Dacă a > b atunci
            returnează (1 / a) * s(a - 1, b, c)
        altfel
            Dacă a < b atunci
                returnează (1 / b) * s(a, b - 1, c)
            altfel
                returnează c * s(a - 1, b - 1, c - 1)
            SfDacă
        SfDacă
    SfDacă
SfSubalgoritm
```

Care trebuie să fie relația dintre ***a***, ***b*** și ***c*** pentru a se obține $1/C_b^a$ (unde C_b^a reprezintă combinări de ***b*** elemente luate câte ***a***)

A. a + b = c
B. a + c = b
C. b - c = a
D. b + c = a - b

UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Concurs de Admitere 19 iulie 2021

Proba scrisă la INFORMATICĂ

BAREM ȘI REZOLVARE

OFICIU: 10 puncte

1	A, C	3 puncte
2	B	3 puncte
3	C	3 puncte
4	D	3 puncte
5	D	3 puncte
6	B,C	3 puncte
7	A	3 puncte
8	B, C	3 puncte
9	A,B,D	3 puncte
10	B, C	3 puncte
11	C, D	3 puncte
12	C	3 puncte
13	D	3 puncte
14	A, B, C, D	3 puncte
15	B	3 puncte
16	B	3 puncte
17	C, D	3 puncte
18	A, C, D	3 puncte
19	A, B, C, D	3 puncte
20	A,C	3 puncte
21	B, D	3 puncte
22	C	3 puncte
23	B, D	3 puncte
24	D	3 puncte
25	B, D	3 puncte
26	A	3 puncte
27	B	3 puncte
28	D	3 puncte
29	B, C	3 puncte
30	B, C	3 puncte