

**În atenția concurenților:**

- Se consideră că indexarea șirurilor începe de la 1.
- Problemele tip grilă din Partea A pot avea unul sau mai multe răspunsuri corecte care trebuie indicate de candidat pe formularul special de pe foaia de concurs. Notarea subiectului de tip grilă se face conform sistemului de punctare parțială din regulamentul concursului.
- Pentru problema din Partea B se cer rezolvări complete scrise pe foaia de concurs, fiind evaluate în detaliu conform baremului.
  - La întrebările B1-B5 se va răspunde în limbaj natural/matematic. Rezolvarea cerinței B6 se va scrie în *pseudocod* sau *Pascal/C/C++*.
  - Primul criteriu în evaluarea rezolvărilor va fi **corectitudinea** algoritmului, iar apoi **performanța** din punct de vedere al  *timpului de executare* și al *spațiului de memorie utilizat*.
  - Este obligatorie descrierea și justificarea** subalgoritmilor **înaintea** rezolvărilor. Se vor scrie, de asemenea, **comentarii** pentru a ușura înțelegerea detaliilor tehnice ale soluției date, a semnificației identificatorilor, a structurilor de date folosite etc. Neîndeplinirea acestor cerințe duce la pierderea a 10% din punctajul aferent subiectului.
  - Nu se vor folosi funcții sau biblioteci predefinite (de exemplu: *STL*, funcții predefinite pe șiruri de caractere).

**Partea A (60 puncte)**

**A.1. Ce se afișează? (6 puncte)**

Se consideră următorul program:

Variantă C	Variantă C++	Variantă Pascal
<pre>#include &lt;stdio.h&gt;  int prelVector(int v[], int *n) {     int s = 0; int i = 2;     while (i &lt;= *n) {         s = s + v[i] - v[i - 1];         if (v[i] == v[i - 1])             *n = *n - 1;         i++;     }     return s; }  int main(){     int v[8];     v[1] = 1; v[2] = 4; v[3] = 2;     v[4] = 3; v[5] = 3; v[6] = 10;     v[7] = 12;     int n = 7;     int rezultat = prelVector(v, &amp;n);     printf("%d;%d", n, rezultat);     return 0; }</pre>	<pre>#include &lt;iostream&gt; using namespace std; int prelVector(int v[], int&amp;n) {     int s = 0; int i = 2;     while (i &lt;= n) {         s = s + v[i] - v[i - 1];         if (v[i] == v[i - 1])             n--;         i++;     }     return s; }  int main(){     int v[8];     v[1] = 1; v[2] = 4; v[3] = 2;     v[4] = 3; v[5] = 3; v[6] = 10;     v[7] = 12;     int n = 7;     int rezultat = prelVector(v, n);     cout &lt;&lt; n &lt;&lt; "; " &lt;&lt; rezultat;     return 0; }</pre>	<pre>type vector=array [1..10] of integer; function prelVector(v: vector;     var n: integer): integer; var s, i: integer; begin     s := 0; i := 2;     while (i &lt;= n) do         begin             s := s + v[i] - v[i - 1];             if (v[i] = v[i - 1]) then                 n := n - 1;             i := i + 1;         end;     prelVector := s; end; var n, rezultat:integer; v:vector; begin     n := 7;     v[1] := 1; v[2] := 4; v[3] := 2;     v[4] := 3; v[5] := 3; v[6] := 10;     v[7] := 12;     rezultat := prelVector(v,n);     write(n, ';', rezultat); end.</pre>

Precizați care este rezultatul afișat în urma executării programului.

A. 7;11

B.6;9

C. 7;9

D. 7;12

**A.2. Expresie logică (6 puncte)**

Precizați care dintre următoarele expresii are valoarea adevărată dacă și numai dacă numărul natural  $n$  este divizibil cu 3 și are ultima cifră 4 sau 6:

A.  $n \text{ DIV } 3 = 0$  și  $(n \text{ MOD } 10 = 4 \text{ sau } n \text{ MOD } 10 = 6)$

B.  $n \text{ MOD } 3 = 0$  și  $(n \text{ MOD } 10 = 4 \text{ sau } n \text{ MOD } 10 = 6)$

C.  $(n \text{ MOD } 3 = 0 \text{ și } n \text{ MOD } 10 = 4)$  sau  $(n \text{ MOD } 3 = 0 \text{ și } n \text{ MOD } 10 = 6)$

D.  $(n \text{ MOD } 3 = 0 \text{ și } n \text{ MOD } 10 = 4)$  sau  $n \text{ MOD } 10 = 6$

**A.3. Ackermann (6 puncte)**

Se consideră numerele naturale  $m$  și  $n$  ( $0 \leq m \leq 10$ ,  $0 \leq n \leq 10$ ) și subalgoritmul  $\text{Ack}(m, n)$  care calculează valoarea funcției Ackermann pentru valorile  $m$  și  $n$ .

```
Subalgoritm Ack(m, n)
Dacă m = 0 atunci
    returnează n + 1
altfel
    Dacă m > 0 și n = 0 atunci
        returnează Ack(m - 1, 1)
    altfel
        returnează Ack(m - 1, Ack(m, n - 1))
SfDacă
SfSubalgoritm
```

Precizați de câte ori se autoapelează subalgoritmul Ack(m, n) prin executarea secvenței de instrucțiuni:

$m \leftarrow 1, n \leftarrow 2$   
 Ack(m, n)

- A. de 7 ori                      B. de 10 ori  
 C. de 5 ori                      D. de același număr de ori ca și în cazul executării secvenței de instrucțiuni

$m \leftarrow 1, n \leftarrow 3$   
 Ack(m, n)

#### A.4. Suma numerelor trunchiate (6 puncte)

Definim operația de *trunchiere* a unui număr natural cu  $k$  cifre  $\overline{c_1 c_2 \dots c_k}$  astfel:  $\text{trunchiere}(\overline{c_1 c_2 \dots c_k}) = \begin{cases} 0, & \text{dacă } k < 2 \\ \overline{c_1 c_2}, & \text{altfel} \end{cases}$ .

Precizați care dintre următorii subalgoritmi calculează *suma trunchierilor* elementelor unui șir  $x$  cu  $n$  numere naturale mai mici decât 1 000 000 ( $n$  – număr natural,  $1 \leq n \leq 1\,000$ )? De exemplu, dacă  $n = 4$  și  $x = (213, 7, 78\,347, 22)$ , atunci suma trunchierilor este  $21 + 0 + 78 + 22 = 121$ .

- A.**

```

Subalgoritm sumăTrunchiată(n, x)
s ← 0
CâtTimp n > 0 execută
    Dacă x[n] > 9 atunci
        CâtTimp x[n] > 99 execută
            x[n] ← x[n] DIV 10
        SfCâtTimp
        s ← s + x[n]
    SfDacă
    n ← n - 1
SfCâtTimp
returnează s
SfSubalgoritm
        
```

**B.**

```

Subalgoritm sumăTrunchiată(n, x)
s ← n
CâtTimp n > 0 execută
    Dacă x[n] > 9 atunci
        CâtTimp x[n] > 99 execută
            x[n] ← x[n] DIV 10
        SfCâtTimp
        s ← s + x[n]
    SfDacă
    n ← n - 1
SfCâtTimp
returnează s
SfSubalgoritm
        
```

**C.**

```

Subalgoritm sumăTrunchiată(n, x)
s ← 0
CâtTimp n > 0 execută
    Dacă x[n] > 9 atunci
        CâtTimp x[n] > 99 execută
            x[n] ← x[n] DIV 10
        SfCâtTimp
        s ← s + x[n]
    SfDacă
    n ← n - 1
SfCâtTimp
returnează s
SfSubalgoritm
        
```

**D.**

```

Subalgoritm sumăTrunchiată(n, x)
s ← 0
CâtTimp x[n] > 99 execută
    x[n] ← x[n] DIV 10
SfCâtTimp
s ← s + x[n]
returnează s
SfSubalgoritm
        
```

#### A.5. Ce valori sunt necesare? (6 puncte)

Se consideră subalgoritmul dif(a, n), unde  $a$  este un șir cu  $n$  numere întregi ( $n$  – număr natural,  $0 < n < 100$ ):

```

Subalgoritm dif(a, n)
Dacă n = 0 atunci
    returnează 0
SfDacă
Dacă |a[n]| MOD 2 = 0 atunci
    returnează dif(a, n - 1) + a[n]
altfel
    returnează dif(a, n - 1) - a[n]
SfDacă
SfSubalgoritm
        
```

{ |a[n]| reprezintă valoarea absolută a lui a[n] }

Precizați pentru care valori ale lui  $n$  și  $a$  subalgoritmul returnează valoarea 0.

- A.  $n = 4$  și  $a = (6, 4, 5, 5)$                       B.  $n = 4$  și  $a = (-6, 5, 4, -7)$   
 C.  $n = 8$  și  $a = (-6, 5, -1, -4, 1, 4, -7, 6)$                       D.  $n = 8$  și  $a = (-6, -3, 0, 1, 2, 3, -1, 4)$

#### A.6. Generare șir de numere speciale (6 puncte)

Fie  $s$  un șir de numere naturale unde elementele  $s_i$  sunt de forma  $s_i = \begin{cases} x & \text{dacă } i = 1 \\ x + 1 & \text{dacă } i = 2, (i = 1, 2, \dots) \\ s_{i-1} @ s_{i-2} & \text{dacă } i > 2 \end{cases}$ . Operatorul @

concatenează cifrele operandului stâng cu cifrele operandului drept, în această ordine (cifre aferente reprezentării în baza 10), iar  $x$  este un număr natural ( $1 \leq x \leq 99$ ). De exemplu, dacă  $x = 3$ , șirul  $s$  va conține valorile 3, 4, 43, 434, 43443, ... . Precizați numărul cifrelor aceluia termen din șirul  $s$  care precede termenul format din  $k$  ( $1 \leq k \leq 30$ ) cifre.

- A. dacă  $x = 15$  și  $k = 6$ , numărul cifrelor termenului aflat în șirul  $s$  în fața termenului format din  $k$  cifre este 5.  
 B. dacă  $x = 2$  și  $k = 8$ , numărul cifrelor termenului aflat în șirul  $s$  în fața termenului format din  $k$  cifre este 5.  
 C. dacă  $x = 14$  și  $k = 26$ , numărul cifrelor termenului aflat în șirul  $s$  în fața termenului format din  $k$  cifre este 16.  
 D. dacă  $x = 5$  și  $k = 13$ , numărul cifrelor termenului aflat în șirul  $s$  în fața termenului format din  $k$  cifre este 10.

### A.7. Permutări circulare (6 puncte)

Fie un șir  $x$  cu  $n$  elemente numere naturale ( $3 \leq n \leq 10\,000$ ) și numărul natural  $k$  ( $1 \leq k < n$ ). Subalgoritmul  $\text{permCirc}(n, k, x)$  ar trebui să genereze permutarea circulară a șirului  $x$  cu  $k$  poziții la stânga (de exemplu, șirul (4, 5, 2, 1, 3) este o permutare circulară cu 2 poziții la stânga pentru șirul (1, 3, 4, 5, 2)). Din păcate subalgoritmul  $\text{permCirc}(n, k, x)$  nu este corect, deoarece pentru anumite valori ale lui  $n$  și  $k$  nu determină rezultat corect.

```

Subalgoritm permCirc(n, k, x)
  c ← k
  Pentru j = 1, c execută
    unde ← j
    nr ← x[unde]
    Pentru i = 1, n / c - 1 execută
      deUnde ← unde + k
      Dacă deUnde > n atunci
        deUnde ← deUnde - n
      SfDacă
        x[unde] ← x[deUnde]
        unde ← deUnde
    SfPentru
    x[unde] ← nr
  SfPentru
SfSubalgoritm

```

Alegeți valorile lui  $n$ ,  $k$  și  $x$  pentru care algoritmul  $\text{permCirc}(n, k, x)$  generează o permutare circulară a șirului  $x$  cu  $k$  poziții la stânga:

- A.  $n = 6, k = 2, x = (1, 2, 3, 4, 5, 6)$
- B.  $n = 8, k = 3, x = (1, 2, 3, 4, 5, 6, 7, 8)$
- C.  $n = 5, k = 3, x = (1, 2, 3, 4, 5)$
- D.  $n = 8, k = 4, x = (1, 2, 3, 4, 5, 6, 7, 8)$

### A.8. Completați (6 puncte)

Se consideră subalgoritmul  $\text{elimInImpar}(n)$ , unde  $n$  este un număr natural,  $1 \leq n \leq 100\,000$ .

```

Subalgoritm elimInImpar(n)
  Dacă n = 0 atunci
    returnează 0
  SfDacă
  Dacă n MOD 2 = 1 atunci
    returnează elimInImpar(n DIV 10)
  SfDacă
  returnează ...
SfSubalgoritm

```

Precizați instrucțiunea care ar trebui scrisă în locul punctelor de suspensie astfel încât algoritmul să aibă ca efect eliminarea cifrelor cu valori impare din numărul  $n$ .

- A.  $\text{elimInImpar}(n \text{ MOD } 10) * 10 + n \text{ DIV } 10$
- B.  $\text{elimInImpar}(n) * 10 + n \text{ MOD } 10$
- C.  $\text{elimInImpar}(n \text{ DIV } 10) * 10 + n \text{ MOD } 10$
- D.  $\text{elimInImpar}((n \text{ DIV } 10) \text{ MOD } 10) * 10$

### A.9. Oare ce face? (6 puncte)

Dreptunghiul cu laturile de lungimi  $m$  și  $n$  ( $m, n$  – numere naturale,  $0 < m < 101, 0 < n < 101$ ) este împărțit în pătrățele cu latura de lungime 1. Se consideră subalgoritmul  $\text{dreptunghi}(m, n)$ :

```

Subalgoritm dreptunghi(m, n)
  d ← m
  c ← n
  CâtTimp d ≠ c execută
    Dacă d > c atunci
      d ← d - c
    altfel
      c ← c - d
  SfDacă
  SfCâtTimp
  returnează m + n - d
SfSubalgoritm

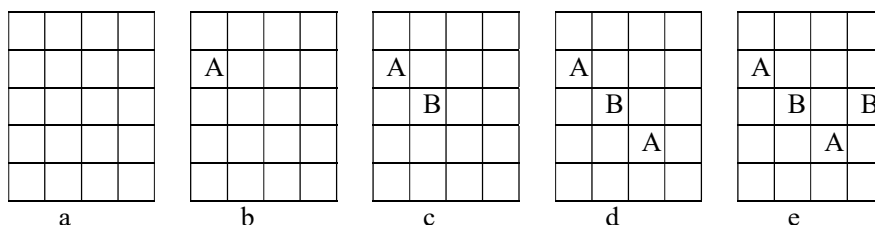
```

Precizați efectul acestui subalgoritm.

- A. Calculează și returnează numărul pătrățelelor cu latura de lungime 1 traversate de o diagonală a dreptunghiului.
- B. Determină în  $d$  cel mai mare divizor comun al laturilor dreptunghiului și returnează diferența dintre suma laturilor dreptunghiului și  $d$ .
- C. Dacă  $m = 8$  și  $n = 12$ , returnează 16.
- D. Dacă  $m = 6$  și  $n = 11$ , returnează 15.

### A.10. Jocul amplasării pieselor de domino pe diagonală (6 puncte)

Fie o tablă dreptunghiulară împărțită în  $n \times m$  căsuțe ( $n$  – numărul liniilor,  $m$  – numărul coloanelor,  $n, m$  – numere naturale,  $2 \leq n \leq 100, 2 \leq m \leq 100$ ). Pe rând, doi jucători A și B execută mutări alternative astfel: la fiecare mutare un jucător hașurează o singură căsuță care este vecină pe diagonală cu căsuța hașurată la pasul anterior de către celălalt jucător și care este nehașurată până în acel moment. Jucătorul care nu mai poate muta, pierde. Jucătorul A face prima mutare, hașurând o căsuță de pe tablă.



Exemplu de tablă de joc: a) inițială ( $n = 5$  și  $m = 4$ ), b) după prima mutare (mutarea lui A), c) după a 2-a mutare (mutarea lui B), d) după a 3-a mutare (mutarea lui A), e) după a 4-a mutare (mutarea lui B)

Determinați condiția în care jucătorul A are strategie sigură de câștig (adică va câștiga jocul, oricare ar fi mutările jucătorului B) și care poate fi prima mutare efectuată de jucătorul A pentru a câștiga jocul.

- A. condiția: numărul  $m$  este impar;  
prima mutare a jucătorului A: o căsuță aflată pe prima linie de sus a tablei (linia 1) și pe o coloană de indice impar
- B. condiția: numărul  $n$  este impar;  
prima mutare a jucătorului A: o căsuță aflată pe o linie de indice par și pe prima coloană din stânga tablei (coloana 1);
- C. condiția: ambele numere  $n$  și  $m$  sunt pare;  
prima mutare a jucătorului A: căsuța din colțul stânga sus (de pe linia 1, coloana 1);
- D. condiția: cel puțin unul dintre numerele  $n$  și  $m$  este impar;  
prima mutare a jucătorului A: căsuța din colțul stânga sus (de pe linia 1, coloana 1).

## Partea B (30 puncte)

### Colivii



În grădina zoologică papagalii trăiesc în colivii numerotate de la 1 la  $n$  ( $1 \leq n \leq 10\,000$ ). La un moment dat, o maimuță jucăușă parcurge și deschide toate coliviile. Speriată de consecințe, se întoarce la prima colivie și închide fiecare a doua colivie (închizând coliviile 2, 4, 6, ...). Maimuței îi place acest joc. De aceea o ia de la început și vizitează fiecare a treia colivie (adică coliviile 3, 6, 9, ...) închizând colivia vizitată dacă o găsește deschisă sau deschizând colivia vizitată dacă o găsește închisă. La a patra parcurgere, vizitează fiecare a patra colivie, procedând similar (schimbând starea coliviilor vizitate). Maimuța repetă jocul, până la ultima parcurgere (a  $n$ -a parcurgere) când închide cea de-a  $n$ -a colivie dacă aceasta este deschisă, sau o deschide, dacă ea este închisă.

### Cerințe:

- B.1. Câte colivii rămân deschise după ultima parcurgere dacă  $n = 10$ ? (2 puncte)
- B.2. Ce numere de ordine au coliviile rămase deschise după ultima parcurgere dacă  $n = 10$ ? (2 puncte)
- B.3. De câte ori este vizitată colivia cu numărul de ordine  $k$  ( $1 \leq k \leq n$ ) în toate cele  $n$  parcurgeri? Justificați răspunsul. (4 puncte)
- B.4. Care este condiția necesară și suficientă astfel încât colivia cu numărul de ordine  $k$  ( $1 \leq k \leq n$ ) să rămână deschisă după ultima parcurgere a celor  $n$  colivii? Justificați răspunsul. (4 puncte)
- B.5. Câte colivii rămân deschise după ultima parcurgere a celor  $n$  colivii? Justificați răspunsul. (4 puncte)
- B.6. Scrieți un subalgoritm care calculează numărul coliviilor rămase deschise (*nrDeschise*) după ultima parcurgere. Numărul coliviilor  $n$  ( $1 \leq n \leq 10\,000$ ) este parametrul de intrare al subalgoritmului, iar *nrDeschise* este parametrul de ieșire. (14 puncte)

*Exemplu 1:* dacă  $n = 5$ , atunci *nrDeschise* = 2 (rămân deschise colivia cu numărul de ordine 1 și colivia cu numărul de ordine 4)

*Exemplu 2:* dacă  $n = 12$ , atunci *nrDeschise* = 3.

### Notă:

1. Toate subiectele sunt obligatorii.
2. Ciornele nu se iau în considerare.
3. Se acordă 10 puncte din oficiu.
4. Timpul efectiv de lucru este de 3 ore și 30 minute.