UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Concurs Mate-Info - varianta 1

Proba scrisă la Informatică

În atenția concurenților:

1. Se consideră că indexarea șirurilor începe de la 1.
2. Problemele tip grilă (Partea A) pot avea unul sau mai multe răspunsuri corecte. Răspunsurile trebuie scrise de candidat pe foaia de concurs (nu pe foaia cu enunțuri). Obținerea punctajului aferent problemei este condiționată de identificarea tuturor variantelor de răspuns corecte și numai a acestora.
3. Pentru problemele din Partea B se cer rezolvări complete pe foaia de concurs.
   - **a.** Rezolvările se vor scrie în *pseudocod* sau *într-un limbaj de programare* (*Pascal/C/C++)*.
   - **b.** 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*.
   - **c.** ***Este obligatorie descrierea și justificarea*** (sub) algoritmilor înaintea rezolvărilor. Se vor scrie, de asemenea, ***comentarii*** pentru a uşura înţelegerea detaliilor tehnice ale soluției date, a semnificaţiei identificatorilor, a structurilor de date folosite etc. Neîndeplinirea acestor cerințe duce la pierderea a 10% din punctajul aferent subiectului.
   - **d.** Nu se vor folosi funcţii sau biblioteci predefinite (de exemplu: *STL*, funcţii predefinite pe şiruri de caractere).

<u>Partea A</u> (30 puncte)

A.1. Oare ce face? (5p)

Se consideră subalgoritmul `alg(x, b)` cu parametrii de intrare două numere naturale ***x*** și ***b*** ($1 \le x \le 1000$, $1 < b \le 10$).

```
Subalgoritm alg(x, b):
   s ← 0
   CâtTimp x > 0 execută
      s ← s + x MOD b
      x ← x DIV b
   SfCâtTimp
   returnează s MOD (b - 1) = 0
SfSubalgoritm
```

Precizați efectul acestui subalgoritm.

a. calculează suma cifrelor reprezentării în baza ***b*** a numărului natural ***x***
b. verifică dacă suma cifrelor reprezentării în baza ***b*** - 1 a numărului ***x*** este divizibilă cu ***b*** - 1
c. verifică dacă numărul natural ***x*** este divizibil cu ***b*** - 1
d. verifică dacă suma cifrelor reprezentării în baza ***b*** a numărului ***x*** este divizibilă cu ***b*** - 1
e. verifică dacă suma cifrelor numărului ***x*** este divizibilă cu ***b*** - 1

A.2. Ce se afișează? (5p)

Se consideră următorul program:

Varianta C++/C

```
int sum(int n, int a[], int s){
   s = 0;
   int i = 1;
   while(i <= n){
      if(a[i] != 0) s += a[i];
      ++i;
   }
   return s;
}


int main(){
   int n = 3, p = 0, a[10];
   a[1] = -1; a[2] = 0; a[3] = 3;
   int s = sum(n, a, p);
   cout << s << ”;“ << p; // printf("%d;%d", s, p);
   return 0;
}
```

Varianta Pascal

```
type vector = array[1..10] of integer;

function sum(n:integer; a:vector; s:integer):integer;
  var i : integer;
  begin
     s := 0; i := 1;
     while (i <= n) do
        begin
           if (a[i] <> 0) then s := s + a[i];
           i := i + 1;
        end;
     sum := s;
  end;
var n, p, s : integer;
        a : vector;
begin
    n := 3; a[1] := -1; a[2] := 0; a[3] := 3; p := 0;
    s := sum(n, a, p);
    writeln(s, ';', p);
end.
```

Care este rezultatul afișat în urma execuției programului?

a. `0;0`
b. `2;0`
c. `2;2`
d. Niciun rezultat nu este corect
e. `0;2`

A.3. Expresie logică (5p)

Se consideră următoarea expresie logică `(X OR Z) AND (NOT X OR Y)`. Alegeți valorile pentru ***X***, ***Y***, ***Z*** astfel încât evaluarea expresiei să dea rezultatul `TRUE`:

```
a. X ← FALSE; Y ← FALSE; Z ← TRUE;
b. X ← TRUE; Y ← FALSE; Z ← FALSE;
c. X ← FALSE; Y ← TRUE; Z ← FALSE;
d. X ← TRUE; Y ← TRUE; Z ← TRUE;
e. X ← FALSE; Y ← FALSE; Z ← FALSE;
```

A.4. Calcul (5p)

Fie subalgoritmul `calcul(a, b)` cu parametrii de intrare ***a*** și ***b*** numere naturale, $1 \leq a \leq 1000$, $1 \leq b \leq 1000$.

```
1.     Subalgoritm calcul(a, b):
2.        Dacă a ≠ 0 atunci
3.           returnează calcul(a DIV 2, b + b) + b * (a MOD 2)
4.        SfDacă
5.        returnează 0
6.     SfSubalgoritm
```

Care din afirmațiile de mai jos sunt false?

- **a.** dacă ***a*** și ***b*** sunt egale, subalgoritmul returnează valoarea lui ***a***
- **b.** dacă ***a*** = 1000 și ***b*** = 2, subalgoritmul se autoapelează de 10 ori
- **c.** valoarea calculată și returnată de subalgoritm este ***a*** / 2 + 2 * ***b***
- **d.** instrucțiunea de pe linia 5 nu se execută niciodată
- **e.** instrucțiunea de pe linia 5 se execută o singură dată

A.5. Identificare element (5p)

Se consideră șirul (1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 2, 5, 11, ...) format astfel: plecând de la șirul numerelor naturale, se înlocuiesc numerele care nu sunt prime cu divizorii lor proprii, fiecare divizor ***d*** fiind considerat o singură dată pentru fiecare număr. Care dintre subalgoritmi determină al ***n***-lea element al acestui șir (***n*** - număr natural, $1 \leq n \leq 1000$)?

a.
```
Subalgoritm identificare(n):
   a ← 1, b ← 1, c ← 1
   CâtTimp c < n execută
      a ← a + 1, b ← a, c ← c + 1, d ← 2
      CâtTimp c ≤ n și d ≤ a DIV 2 execută
         Dacă a MOD d = 0 atunci
            c ← c + 1, b ← d
         SfDacă
         d ← d + 1
      SfCâtTimp
   SfCâtTimp
   returnează b
SfSubalgoritm
```

b.
```
Subalgoritm identificare(n):
   a ← 1, b ← 1, c ← 1
   CâtTimp c < n execută
      c ← c + 1, d ← 2
      CâtTimp c ≤ n și d ≤ a DIV 2 execută
         Dacă a MOD d = 0 atunci
            c ← c + 1, b ← d
         SfDacă
         d ← d + 1
      SfCâtTimp
      a ← a + 1, b ← a
   SfCâtTimp
   returnează b
SfSubalgoritm
```

c.
```
Subalgoritm identificare(n):
   a ← 1, b ← 1, c ← 1
   CâtTimp c < n execută
      a ← a + 1, d ← 2
      CâtTimp c < n și d ≤ a execută
         Dacă a MOD d = 0 atunci
            c ← c + 1, b ← d
         SfDacă
         d ← d + 1
      SfCâtTimp
   SfCâtTimp
   returnează b
SfSubalgoritm
```

d.
```
Subalgoritm identificare(n):
   a ← 1, b ← 1, c ← 1
   CâtTimp c < n execută
      b ← a, a ← a + 1, c ← c + 1, d ← 2
      CâtTimp c ≤ n și d ≤ a DIV 2 execută
         Dacă a MOD d = 0 atunci
            c ← c + 1, b ← d
         SfDacă
         d ← d + 1
      SfCâtTimp
   SfCâtTimp
   returnează b
SfSubalgoritm
```

e.
```
Subalgoritm identificare(n):
   a ← 1, b ← 1, c ← 1
   CâtTimp c < n execută
      a ← a + 1, b ← a, c ← c + 1, d ← 2
      f ← false
      CâtTimp c ≤ n și d ≤ a DIV 2 execută
         Dacă a MOD d = 0 atunci
            c ← c + 1, b ← d, f ← true
         SfDacă
         d ← d + 1
      SfCâtTimp
      Dacă f atunci
         c ← c - 1
      SfDacă
   SfCâtTimp
   returnează b
SfSubalgoritm
```

A.6. Factori primi (5p)

Fie subalgoritmul `factoriPrimi(n, d, k, x)` care determină cei ***k*** factori primi ai unui număr natural ***n***, începând căutarea factorilor primi de la valoarea ***d***. Parametrii de intrare sunt numerele naturale ***n***, ***d*** și ***k***, iar parametrii de ieșire sunt șirul ***x*** cu cei ***k*** factori primi ($1 \leq n \leq 10000$, $2 \leq d \leq 10000$, $0 \leq k \leq 10000$).

```
Subalgoritm factoriPrimi(n, d, k, x):
   Dacă n MOD d = 0 atunci
      k ← k + 1
      x[k] ← d
   SfDacă
   CâtTimp n MOD d = 0 execută
      n ← n DIV d
   SfCâtTimp
   Dacă n > 1 atunci
      factoriPrimi(n, d + 1, k, x)
   SfDacă
SfSubalgoritm
```

Stabiliți de câte ori se autoapelează subalgoritmul `factoriPrimi(n, d, k, x)` în următoarea secvență de instrucțiuni:

```
n ← 120
d ← 2
k ← 0
factoriPrimi(n, d, k, x)
```

a. de 3 ori
b. de 5 ori
c. de 9 ori
d. de 6 ori
e. de același număr de ori ca și în cadrul secvenței de instrucțiuni:

```
n ← 750
d ← 2
k ← 0
factoriPrimi(n, d, k, x)
```

<u>Partea B</u> (60 puncte)

B.1. Conversie (10 puncte)

Fie subalgoritmul `conversie(s, lung)` care transformă un șir ***s*** de caractere, exprimând un număr în baza 16, în numărul corespunzător scris în baza 10. Șirul ***s*** este format din ***lung*** caractere care pot avea ca valori doar cifrele '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' și literele mari 'A', 'B', 'C', 'D', 'E', 'F' (***lung*** este număr natural, $1 \leq$ ***lung*** ≤ 10).

Scrieți o variantă *recursivă* a subalgoritmului `conversie(s, lung)` care are același antet și același efect cu acesta.

```
Subalgoritm conversie(s, lung):
    nr ← 0
    Pentru i ← 1, lung execută
       Dacă s[i] ≥ 'A' atunci
          nr ← nr * 16 + s[i] - 'A' + 10
       altfel
          nr ← nr * 16 + s[i] - '0'
       SfDacă
    SfPentru
    returnează nr
SfSubalgoritm
```

B.2. Cifre identice (20 puncte)

Se consideră două numere naturale ***a*** și ***b***, unde $1 \leq a \leq 1\,000\,000$ și $1 \leq b \leq 1\,000\,000$.

Scrieți un subalgoritm care determină șirul ***x***, având ***k*** elemente (***k*** - număr natural, $0 \leq k \leq 1000$), format din toate numerele naturale cuprinse în intervalul [***a***, ***b***] care au toate cifrele identice. Dacă nu există astfel de numere, ***k*** va fi 0. Numerele ***a*** și ***b*** sunt parametrii de intrare ai subalgoritmului, iar ***k*** și ***x*** vor fi parametrii de ieșire.

Exemplul 1: dacă ***a*** = 8 și ***b*** = 120, atunci ***k*** = 12 și ***x*** = (8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 111).
Exemplul 2: dacă ***a*** = 590 și ***b*** = 623, atunci ***k*** = 0 și ***x*** este șirul vid.

B.3. Roboțel plimbăreț (30 puncte)

Un roboțel se poate plimba pe o hartă dată sub forma unei matrice pătratice de dimensiune impară, lăsând în fiecare celulă a hărții un anumit număr de obiecte. Plimbarea roboțelului se desfășoară după următoarele reguli:

- în celula din care pornește roboțelul lasă un obiect, în a doua celulă în care ajunge lasă 2 obiecte, în a treia celulă în care ajunge lasă 3 obiecte, ș.a.m.d.;
- roboțelul pornește din mijlocul ultimei coloane și merge întotdeauna un pas pe diagonală în celula alăturată de sus-dreapta (direcție paralelă cu diagonala secundară) dacă această celulă există și ea este liberă; dacă celula nu există, atunci:
  - dacă roboțelul se află pe ultima coloană, atunci el ”sare” în celula aflată pe coloana întâi și linia de deasupra lui dacă aceasta este liberă;
  - dacă roboțelul se află pe prima linie, el ”sare” în celula aflată pe ultima linie și coloana din dreapta lui dacă aceasta este liberă;
  - dacă roboțelul se află în colțul dreapta-sus al hărții, el ”sare” în celula aflată pe ultima linie și prima coloană dacă aceasta este liberă.
- în situația în care celula pe care trebuie să ajungă nu este liberă, roboțelul face un pas la stânga în celula alăturată de pe aceeași linie cu el.

Aceste reguli asigură vizitarea o singură dată a tuturor celulelor din hartă (și, implicit, evitarea blocajelor în deplasare). După ce roboțelul lasă obiecte în toate celulele hărții, el se oprește.

De exemplu, pentru o hartă cu 5 × 5 celule, primii 22 pași efectuați de roboțel ar fi:

9	3	22	16	15
2	21	20	14	8
	19	13	7	1
18	12	6	5	
11	10	4		17

Scrieți un subalgoritm care determină numărul de obiecte ***nr*** lăsate de roboțel în celulele de pe diagonala principală a hărții. Parametrul de intrare al subalgoritmului este dimensiunea hărții ***n*** (***n*** - număr natural impar, 3 ≤ ***n*** ≤ 100), iar ***nr*** va fi parametrul de ieșire (***nr*** - număr natural).

Exemplu 1: dacă ***n*** = 5, atunci ***nr*** = 65.
Exemplu 2: dacă ***n*** = 11, atunci ***nr*** = 671.

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.

BAREM
OFICIU .... 10 puncte

Partea A .... 30 puncte
A. 1. c, d.. **5 puncte**
A. 2. b .... **5 puncte**
A. 3. a, d.. **5 puncte**
A. 4. a, c, d .... **5 puncte**
A. 5. e.. **5 puncte**
A. 6. a, e .... **5 puncte**

Partea B .... 60 puncte
B. 1. Conversie.. 10 puncte
– respectarea antetului .... 2 puncte
– condiția de oprire din recursivitate .... 1 puncte
– valoarea returnată la oprirea recursivității .... 1 puncte
– condiția pentru caracter diferit de cifră.. 2 puncte
– valoarea returnată atunci când condiția pentru caracter diferit de cifră este îndeplinită.......... 2 puncte
– valoarea returnată atunci când condiția pentru caracter diferit de cifră nu este îndeplinită...... 2 puncte

```
Subalgoritm conversie (s, lung):
    Dacă lung > 0 atunci
       Dacă s[lung] ≥ 'A' atunci
          returnează conversie(s, lung - 1) * 16 + s[lung] - 'A' + 10
       altfel
          returnează conversie(s, lung - 1) * 16 + s[lung] - '0'
       SfDacă
    altfel
       returnează 0
    SfDacă
 SfSubalgoritm
```

B. 2. Cifre identice .... 20 puncte

- V1: **generarea** numerelor având cifre identice **pe bază de calcule** .... 20 puncte
  - respectarea parametrilor de intrare și ieșire .... 2 puncte
  - generarea numerelor (cu o singură cifră și/sau ca multipli de 11, 111, 1111, ...)............ 16 puncte
  - salvarea numerelor cu cifre identice în vector .... 2 puncte
- V2: **considerarea** și **verificarea** tuturor numerelor din intervalul [***a***, ***b***] .... 10 puncte
  - respectarea parametrilor de intrare și ieșire .... 2 puncte
  - verificarea dacă un număr are toate cifrele identice .... 4 puncte
  - parcurgerea și verificarea tuturor numerelor din [***a***, ***b***].. 2 puncte
  - salvarea numerelor cu cifre identice în vector .... 2 puncte

B. 3. Roboțel plimbăreț.. 30 puncte

- V1: determinarea corectă a valorii prin calcule ***(n*n*n+n)/2*** .... 30 puncte
  - respectarea parametrilor de intrare și ieșire .... 2 puncte
  - calcul.. 14 puncte
  - prezentarea detaliată a metodei de obținere a formulei de calcul .... 14 puncte
- V2: determinarea corectă a valorii prin simulare.. 25 puncte
  - respectarea parametrilor de intrare și ieșire .... 2 puncte
  - parcurgerea celor *n* x *n* celule .... 4 puncte
  - deplasarea corectă (în cele 4 cazuri posibile) .... 4x4 puncte
  - calculul sumei elementelor de pe diagonală .... 3 puncte

Partea B (soluții) .... 60 puncte
B. 1. Conversie.. 10 puncte

```
Subalgoritm conversie(s, lung):
    Dacă lung > 0 atunci
       Dacă s[lung] ≥ 'A' atunci
          returnează conversie(s, lung - 1) * 16 + s[lung] - 'A' + 10
       altfel
          returnează conversie(s, lung - 1) * 16 + s[lung] - '0'
       SfDacă
    altfel
       returnează 0
    SfDacă
 SfSubalgoritm
```

B. 2. Cifre identice .... 20 puncte

- generarea numerelor având cifre identice pe bază de calcule .... 20 puncte

```
void cifreIdentice(int a, int b, int &k, int x[]){
        k = 0;
        for (int i = a; ((i < 10) && (i <= b)); i++){
                x[++k] = i;
        }
        int crt = 11;
        int ratie = 11;
        while (crt <= b){
                if (crt >= a)
                        x[++k] = crt;
                crt = crt + ratie;
                if (crt > 9 * ratie){
                        crt = ratie * 10 + 1;
                        ratie = crt;
                }
        }
}
```

B. 3. Roboțel plimbăreț .... 30 puncte

- determinarea corectă a valorii prin calcule (***n*n*n+n)/2*** .... 30 puncte

				24		
			23	**17**		10
9		22	16	15	**9**	3
	21	20	14	8	2	
25	19	**13**	7	1		
18	12	6	**5**			
11		4		17		

Pentru cazul cu ***n*** mai mare ca 3, se formează *n* diagonale:

- pe prima diagonală se află numerele 1, 2, ..., *n*, iar în suma noastră îl considerăm pe ultimul (*n*)
- pe a 2-a diagonală se află numerele $n + 1$, $n + 2$, ..., $2n$, iar în suma noastră îl considerăm pe penultimul ($2n - 1$)
- pe a 3-a diagonală se află numerele $2n + 1$, $2n + 2$, ..., $3n$, iar în suma noastră îl considerăm pe antepenultimul ($3n - 2$)
- ...
- pe a *n*-a diagonală se află numerele $(n - 1) * n + 1$, ..., $n * n$, iar în suma noastră îl considerăm pe primul $(n - 1) * n + 1$ (sau $n * n - (n - 1)$)

Deci suma elementelor de pe diagonala principală va fi: $n + (2n - 1) + (3n - 2) + ... + n * n - (n - 1) =$
$n + 2n + 3n + ... + n * n - (1 + 2 + ... + (n - 1)) = n * (1 + 2 + ... + n) - (n - 1) * n / 2 = n * n * (n + 1) / 2 - (n - 1) * n / 2 = (n * n * n + n * n - n * n + n) / 2 = (n * n * n + n) / 2$
Dacă $n = 5$, suma va fi 65

```
int obiecte(int n){
        return (n * n * n + n) / 2;
}
```