UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Concurs Mate-Info - model

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 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.
   - **d.** Nu se vor folosi funcții sau biblioteci predefinite (de exemplu: *STL*, funcții predefinite pe șiruri de caractere).

Partea A (60 puncte)

A.1. Oare ce face? (6 puncte)

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

A.2. Ce se afișează? (6 puncte)

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

A.3. Expresie logică (6 puncte)

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;

A.4. Calcul (6 puncte)

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

A.5. Identificare element (6 puncte)

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
      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
```

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
```

A.6. Factori primi (6 puncte)

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)` prin execuția următoarei secvențe 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 același număr de ori ca și în cadrul secvenței de instrucțiuni:

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

A.7. Oare ce face? (6 puncte)

Se consideră subalgoritmul `expresie(n)`, unde ***n*** este un număr natural ($1 \leq n \leq 10000$).

```
Subalgoritm expresie(n):
   Dacă n > 0 atunci
      Dacă n MOD 2 = 0 atunci
         returnează -n * (n + 1) + expresie(n - 1)
      altfel
         returnează n * (n + 1) + expresie(n - 1)
      SfDacă
   altfel
      returnează 0
   SfDacă
SfSubalgoritm
```

Precizați forma matematică a expresiei *E(n)* calculată de acest subalgoritm:

A. $E(n) = 1* 2 - 2 * 3 + 3 * 4 + ... + (-1)^{n+1} * n * (n + 1)$
B. $E(n) = 1* 2 - 2 * 3 + 3 * 4 + ... + (-1)^{n} * n * (n + 1)$
C. $E(n) = 1* 2 + 2 * 3 + 3 * 4 + ... + (-1)^{n+1} * n * (n + 1)$
D. $E(n) = 1* 2 + 2 * 3 + 3 * 4 + ... + (-1)^{n} * n * (n + 1)$

A.8. Expresie logică (6 puncte)

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

A. ***X*** ← *FALSE*; ***Y*** ← *FALSE*; ***Z*** ← *FALSE*;
B. ***X*** ← *FALSE*; ***Y*** ← *FALSE*; ***Z*** ← *TRUE*;
C. ***X*** ← *FALSE*; ***Y*** ← *TRUE*; ***Z*** ← *FALSE*;
D. ***X*** ← *TRUE*; ***Y*** ← *FALSE*; ***Z*** ← *TRUE*;

A.9. Valori ale parametrului de intrare (6 puncte)

Se consideră următorul subalgoritm:

```
Subalgoritm SA9(a):
        Dacă a < 50 atunci
                Dacă a MOD 3 = 0 atunci
                        returnează SA9(2 * a - 3)
                altfel
                        returnează SA9(2 * a - 1)
              SfDacă
        altfel
                returnează a
        SfDacă
SfSubalgoritm
```

Pentru care dintre valorile parametrului de intrare ***a*** subalgoritmul va returna valoarea 61?

A. 16

B. 61
C. 4
D. 31

A.10. Instrucțiuni lipsă (6 puncte)

Se dă următorul subalgoritm:

```
1:    Subalgoritm cautare(x, n, val):
2:          Dacă n = 1 atunci
3:                returnează (x[1] = val)
4:          altfel
5:                returnează cautare(x, n - 1, val)
6:          SfDacă
7:    SfSubalgoritm
```

Ce instrucțiune sau instrucțiuni trebuie adăugate și unde astfel încât în urma apelului, subalgoritmul `cautare(x, n, val)` să determine dacă elementul ***val*** face sau nu parte din șirul ***x*** cu ***n*** elemente (***n*** număr natural strict mai mare ca zero)?

A. Linia 5 trebuie modificată în: `returnează ((x[n] = val) și cautare(x - 1, n, val))`
B. Linia 5 trebuie modificată în: `returnează ((x[n] = val) sau cautare(x, n - 1, val))`
C. Linia 5 trebuie modificată în: `dacă (x[n] = val) atunci returnează true altfel returnează cautare(x, n - 1, val)`
D. nu trebuie modificată nici o instrucțiune

Partea B (30 puncte)

1. Prefix (15 puncte)

Cifra de control a unui număr natural se determină calculând suma cifrelor numărului, apoi suma cifrelor sumei și așa mai departe până când suma obținută reprezintă un număr cu o singură cifră. De exemplu, *cifra de control* a numărului 182 este 2 (1 + 8 + 2 = 11, 1 + 1 = 2).

Un număr ***p*** format din exact ***k*** cifre este *prefix* al unui număr ***q*** cu cel puțin ***k*** cifre dacă numărul format din primele ***k*** cifre ale numărului ***q*** (parcurse de la stânga la dreapta) este egal cu ***p***. De exemplu, 17 este prefix al lui 174, iar 1713 este prefix al lui 1 713 242.

Se consideră un număr ***nr*** natural ($0 < nr \leq 30\,000$) și o matrice (un tablou bidimensional) ***A*** cu ***m*** linii și ***n*** coloane ($0 < m \leq 100$, $0 < n \leq 100$), având ca elemente numere naturale mai mici decât 30 000. Se consideră și subalgoritmul `cifrăControl(x)` pentru determinarea cifrei de control asociată numărului ***x***:

```
Subalgoritm cifraControl(x):
    s ← 0
   CâtTimp x > 0 execută
      s ← s + x MOD 10
      x ← x DIV 10
      Dacă x = 0 atunci
         Dacă s < 10 atunci
            Returnează s
         altfel
            x ← s
            s ← 0
         SfDacă
      SfDacă
   SfCâtTimp
   returnează s
SfSubalgoritm
```

Cerințe:

a. Scrieți o variantă *recursivă* (fără structuri repetitive) a subalgoritmului `cifrăControl(x)` care are același antet și același efect cu acesta. (**5 puncte**)
b. Scrieți modelul matematic al variantei recursive a subalgoritmului `cifrăControl(x)` (dezvoltat la punctul a). (**3 puncte**)
c. Scrieți un subalgoritm care, folosind subalgoritmul `cifrăControl(x)`, determină cel mai lung prefix (notat ***prefix***) al numărului ***nr*** care se poate construi folosind cifrele de control corespunzătoare elementelor din matricea dată. O astfel de cifră de control poate fi folosită de oricâte ori în construirea prefixului. Dacă nu se poate construi un prefix, ***prefix*** va fi -1. Parametrii de intrare ai subalgoritmului sunt numerele ***nr***, ***m***, ***n***, matricea ***A***, iar parametrul de ieșire este ***prefix***. (**7 puncte**)

Exemplu*:** dacă avem ***nr = 12319, ***m*** = 3 și ***n*** = 4 și matricea $A = \begin{pmatrix} 182 & 12 & 274 & 22 \\ 22 & 1 & 98 & 56 \\ 5 & 301 & 51 & 94 \end{pmatrix}$,

cel mai lung prefix este ***prefix*** = 1231, cifrele de control fiind:

Element din matrice	182	12	274	22	1	98	56	5	301	51	94
Cifră control	2	3	4	4	1	8	2	5	4	6	4

2. Robi-grădina (15 puncte)

Un grădinar pasionat de tehnologie decide să folosească o „armată" de roboți pentru a uda straturile din grădina sa. El dorește să folosească apă de la izvorul situat la capătul cărării principale care străbate grădina. Fiecărui strat îi este asociat un robot și fiecare robot are de udat un singur strat. Toți roboții pornesc de la izvor în misiunea de udare a straturilor la aceeași oră a dimineții (spre exemplu 5:00:00) și lucrează în paralel și neîncetat un același interval de timp. Ei parcurg cărarea principală până la stratul lor, pe care îl udă și revin la izvor pentru a-și reumple rezervorul de apă. La finalul intervalului de timp aferent activității, toți roboții se opresc simultan, indiferent de starea lor curentă. Inițial, la izvor este amplasat un singur robinet. Grădinarul constată însă că apar întârzieri în programul de udare a plantelor deoarece roboții trebuie să aștepte la rând pentru reumplerea rezervoarelor cu apă, astfel încât consideră că cea mai bună soluție este să instaleze mai multe robinete pentru alimentarea roboților. Roboții pornesc dimineața cu rezervoarele umplute. Doi roboți nu își pot umple rezervorul în acelaşi moment de la acelaşi robinet.

Se cunosc: intervalul de timp t cât cei n roboți lucrează (exprimat în secunde), numărul de secunde d_i necesare pentru a parcurge distanța de la izvor la stratul asociat, numărul de secunde u_i necesar pentru udarea acestui strat și faptul că umplerea rezervorului propriu cu apă durează exact o secundă pentru fiecare robot (t, n, d_i, u_i - numere naturale, $1 \le t \le 20000$, $1 \le n \le 100$, $1 \le d_i \le 1000$, $1 \le u_i \le 1000$, $i = 1, 2, ..., n$).

Cerințe:

a. Enumerați roboții care se întâlnesc la izvor la un anumit moment de timp mt ($1 \le mt \le t$). Justificați răspunsul. Notă: roboții se identifică prin numărul lor de ordine. (**3 puncte**)

b. Care este numărul minim de robinete suplimentare ***minRobineteSuplim*** pe care trebuie să le instaleze grădinarul astfel încât roboții să nu aștepte deloc unul după altul pentru reumplerea rezervorului? Justificați răspunsul. (2 **puncte**)

c. Scrieți un subalgoritm care determină numărul minim de robinete suplimentare ***minRobineteSuplim***. Parametrii de intrare sunt numerele n și t, șirurile d și u cu câte n elemente fiecare, iar parametrul de ieșire este ***minRobineteSuplim***. (**10 puncte**)

Exemplu 1: dacă $t = 32$, $n = 5$, $d = (1, 2, 1, 2, 1)$, $u = (1, 3, 2, 1, 3)$ atunci ***minRobineteSuplim*** = 3. Explicație: robotul care se ocupă de stratul 1 are nevoie de o secundă pentru a ajunge la strat, o secundă pentru a uda stratul și de încă o secundă pentru a se întoarce la izvor; el se întoarce la izvor pentru a-și reumple rezervorul după 1 + 1 + 1 = 3 secunde de la ora de plecare (5:00:00), deci la ora 5:00:03; el își reumple rezervorul într-o secundă și pornește înapoi spre strat la ora 5:00:04; revine la ora 5:00:07 pentru a-și reumple rezervorul, continuând ritmul de udare a straturilor, ș.a.m.d.; deci, primul, al doilea, al patrulea și al cincilea robot se întâlnesc la izvor la ora 05:00:23; în consecință, este nevoie de 3 robinete suplimentare.

Exemplu 2: dacă $t = 37$, $n = 3$, $d = (1, 2, 1)$, $u = (1, 3, 2)$, atunci ***minRobineteSuplim*** = 1.

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 3.5 ore.

BAREM & REZOLVĂRI
OFICIU ..... 10 puncte

Partea A ..... 60 puncte
A. 1. Oare ce face? Răspuns: C, D ..... **6 puncte**
A. 2. Ce se afișsează? Răspuns: B ..... **6 puncte**
A. 3. Expresie logică. Răspuns: A, D ..... **6 puncte**
A. 4. Calcul. Răspuns: **A, C,** D ..... **6 puncte**
A. 5. Identificare element. Răspuns: A ..... **6 puncte**
A. 6. Factori primi. Răspuns: A, D ..... **6 puncte**
A. 7. Oare ce face? Răspuns: A ..... **6 puncte**
A. 8. Expresie logică. Răspuns: A, B, D ..... **6 puncte**
A.9. Valoari ale parametrului de intrare. Răspuns: A, B, D ..... **6 puncte**
A.10. Instrucțiune lipsă. Răspuns: B, C ..... **6 puncte**

Partea B ..... 30 puncte
B. 1. Prefix ..... 15 puncte
B.1.a. versiunea recursivă a subalgoritmului `cifrăControl(x)` ..... **5 puncte**
– respectarea antetului ..... 1 punct
– condiție oprire recursivitate ..... 1 punct
– autoapel (logica, parametrii) ..... 2 puncte
– valoari returnate ..... 1 punct

```
Subalgoritm cifraControl(x):
   Dacă x > 9 atunci
      s ← x MOD 10 + cifraControl(x DIV 10)
      Dacă s < 10 atunci
         Returnează s
      altfel
         Returnează cifraControl(s)
      SfDacă
   altfel
      Returnează x
   SfDacă
SfSubalgoritm
```

B.1.b. modelul matematic pentru `cifrăControl(x)` ..... **3 puncte**

$$cifraControl(x) = \begin{cases} x, & dacă\ x < 10 \\ cifraControl(x\ MOD\ 10) + cifraControl(x\ DIV\ 10)), & altfel \end{cases}$$

B.1.c. subalgoritmul pentru cel mai lung prefix ..... 7 puncte

Varianta 1: matricea se parcurgere o singură dată și se construiește un vector de apariții pentru cifrele de control corespunzătoare elementelor din matrice. Se reține într-un vector cifrele numărului dat (*nr*). Se parcurge vectorul acestor cifre (începând cu cifra cea mai semnificativă) și se verifică apariția cifrelor în vectorul de apariții construit anterior ..... **7 puncte**

Notă: aceeași idee se poate rezolva cu vector de frecvență al cifrelor (în locul vectorului de apariții)

```
const int NRMAXCIFRE = 10;
int prefixMaxCifre(int nr, int m, int n, int A[][MAX]){
   bool aparitii[NRMAXCIFRE];   int cifre[MAX];
   int nrCifre; int i = 0;
   for (i = 0; i < NRMAXCIFRE; i++)                    // inițializarea vectorului de frecvențe
      aparitii[i] = 0;
   for (i = 0; i < m; i++){
      for (int j = 0; j < n; j++){
         aparitii[cifraControl(A[i][j])] = 1;                    // apariția cifrei de control
      }
   }
   nrCifre = 0;                                          // numărul cifrelor numărului nr dat
   while (nr > 0){
      cifre[nrCifre++] = nr % 10;                // elementele șirului cifrelor numărului nr dat
```

```
      nr /= 10;
   }
   i = nrCifre - 1;                                          // parcurgem șirul cifrelor
   while ((i >= 0) && (aparitii[cifre[i]]))       // există cifră de control = cu cifra curentă
      i--;                                                                  // din nr
   return nrCifre - i - 1;                                  // lungimea celui mai lung prefix
}
```

Varianta 2: Se reține într-un vector cifrele numărului dat (*nr*). Se caută fiecare cifră a numărului în matrice (matricea se parcurgere, astfel, de mai multe ori) ..... **5 puncte**

```
bool cautaCifra(int cifra, int m, int n, int A[][MAX]){
   for(int i = 0; i < m; i++){
      for(int j = 0; j < n; j++){
         if(cifra == cifraControl(A[i][j]))// în matricea cifrelor de control există cifra căutată
            return true;
      }
   }
   return false;                                                // nu am găsit cifra căutată
}
int prefixMaxCifre_v2(int nr, int m, int n, int A[][MAX]){
   int cifre[MAX];
   int nrCifre = 0;                                       // numărul cifrelor numărului nr dat
   while(nr > 0){
      cifre[nrCifre++] = nr % 10;               // elementele șirului cifrelor numărului nr dat
      nr /= 10;
   }
   int i = nrCifre - 1;                                          // parcurgem șirul cifrelor
   int p = 0;
   while(i >= 0){
      if(cautaCifra(cifre[i], m, n, A)){              // căutăm cifrele numărului în matricea A
         p++;
         i--;
      }
      else
         return p;
   }
   return p;     // numărul cifrelor consecutive din nr găsite în matrice (lungimea prefixului)
}
```

B. 2. Robi grădină ..... **15 puncte**

dacă $n = 5$, $d = (1, 2, 1, 2, 1)$, $u = (1, 3, 2, 1, 3)$, $t = 32$, se calculează valorile lui q:

2 * 1 + 1 + 1 = 4, 2* 2 + 3 + 1 = 8, 2 * 1 + 2 + 1 = 5, 2 * 2 + 1 + 1 = 6, 2 * 1 + 3 + 1 = 6 => (4, 8, 5, 6, 6)

Se construiește un vector v cu $t = 32$ valori nule. Se consideră, pe rând, valoarea lui q pentru fiecare roboțel și se incrementează în vector căsuța corespunzătoare multiplilor lui q.

$q = 4$

	1	2	3	**4**	5	6	7	**8**	9	10	11	**12**	13	14	15	**16**	17	18	19	**20**
v	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**

	21	22	23	**24**	25	26	27	**28**	29	30	31	**32**
v	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**

$q = 8$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	0	0	0	1	0	0	0	2	0	0	0	1	0	0	0	2	0	0	0	1

	21	22	23	24	25	26	27	28	29	30	31	32
v	0	0	0	2	0	0	0	1	0	0	0	2

$q = 5$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	0	0	0	1	1	0	0	2	0	1	0	1	0	0	1	2	0	0	0	2

	21	22	23	24	25	26	27	28	29	30	31	32
v	0	0	0	2	1	0	0	1	0	1	0	2

$q = 6$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	0	0	0	1	1	1	0	2	0	1	0	2	0	0	1	2	0	1	0	2

	21	22	23	24	25	26	27	28	29	30	31	32

v	0	0	0	3	1	0	0	1	0	2	0	2

q = 6

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	0	0	0	1	1	2	0	2	0	1	0	3	0	0	1	2	0	2	0	2

	21	22	23	24	25	26	27	28	29	30	31	32
v	0	0	0	4	1	0	0	1	0	3	0	2

Se determină maximul din vectorul ***v***. În cazul curent, maximul din șir este 4 (la momentul 24), deci mai trebuie 4 - 1 = 3 robinete suplimentare.

B.2.a. la un anumit moment de timp ***mt*** ($1 \leq mt \leq t$) se întâlnesc la izvor roboții care au valoarea ***q*** (egală cu suma dintre timpul necesar deplasării până la strat și înapoi, timpul necesar udării stratului și timpul necesar umplerii rezervorului) multiplu de ***mt***..... 3 puncte

B.2.b. numărul minim de robinete suplimentare este egal cu maximul vectorului ***v*** - 1, unde vectorul v reține, pentru fiecare moment de timp, câți roboți se întâlnesc la izvor în momentul respectiv ..... 2 puncte

B.2.c. Dezvoltare subalgoritm

- V1: folosirea unui vector de frecvență pentru multiplii timpilor de lucru ai fiecărui robot..... 10 puncte
  - c.1. respectarea parametrilor de intrare și ieșire..... **2 puncte**
  - c.2. calcul timp de lucru (***q*** = 2 * deplasare + udare + încărcare) ..... **1 punct**
  - c.3. prelucrarea vectorului de frecvențe ..... **4 puncte**
    - c.3.1. inițializare vector ..... 1 punct
    - c.3.2. update frecvență..... 3 puncte
      1. v1a sau v1b: parcurgere din q în q..... 3 puncte
      2. v2a sau v2b: parcurgere din 1 în 1 și verificare multiplu de q..... 2 puncte
  - c.4. stabilirea numărului maxim de robinete ..... 2 **puncte**
      3. deodată cu incrementarea sau după terminarea incrementării ..... 1 punct
  - c.5. determinarea numărului de robinete suplimentare (max - 1)..... **1 punct**

```
int robiV1a(int n, int d[], int u[] int t){
    int aux[200000];
    int max = 1;
    for (int i = 1; i <= t; i++)
         aux[i] = 0;
    for (int j = 1; j <= n; j++){
         int q = d[j] * 2 + u[j] + 1;
         for (int i = q; i <= t; i = i + q){
                 aux[i]++;
                 if (max < aux[i])
                         max = aux[i];
         }       //for i
    }    //for j
    return max - 1;
}
```

```
int robiV1b(int n, int d[], int u[], int t){
    int aux[200000];
    int max = 1;
    for (int i = 1; i <= t; i++)
         aux[i] = 0;
    for (int j = 1; j <= n; j++){
         int q = d[j] * 2 + u[j] + 1;
         for (int i = q; i <= t; i = i + q){
                 aux[i]++;
         }       //for i
    }    //for j
    for (int i = 1; i <= t; i++){
         if (max < aux[i])
                 max = aux[i];
    }    //for i
    return max - 1;
}
```

```
int robiV2a(int n, int d[], int u[], int t){
    int aux[200000];
    int max = 1;
    for (int i = 1; i <= t; i++)
         aux[i] = 0;
    for (int j = 1; j <= n; j++){
         int q = d[j] * 2 + u[j] + 1;
         for (int i = q; i <= t; i = i + 1){
                 if (i % q == 0){
                         aux[i]++;
                         if (max < aux[i])
                                 max = aux[i];
                 }       //if (i % q)
         }       //for i
    }    //for j
    return max - 1;
}
```

```
int robiV2b(int n, int d[], int u[], int t){
    int aux[200000];
    int max = 1;
    for (int i = 1; i <= t; i++)
         aux[i] = 0;
    for (int j = 1; j <= n; j++){
         int q = d[j] * 2 + u[j] + 1;
         for (int i = q; i <= t; i = i + 1){
                 if (i % q == 0){
                         aux[i]++;
                 }       //if (i % q)
         }       //for i
    }    //for j
    for (int i = 1; i <= t; i++){
         if (max < aux[i])
                 max = aux[i];
    }    //for i
    return max - 1;
}
```

- V2: simulare..... 8 puncte
  - c.1. respectarea parametrilor de intrare și ieșire ..... 2 puncte

c.2. calcul timp de lucru (***q*** = 2 * deplasare + udare + încărcare) ..... 1 punct
c.3. structura repetitivă pentru timp ..... 0.5 puncte
c.4. structura repetitivă pentru roboți ..... 0.5 puncte
c.5. stabilirea numărului de robinete necesare la un anumit moment de timp ..... 1 punct
c.6 stabilirea numărului maxim de robinete ..... 2 puncte
c.7. determinarea numărului de robinete suplimentare ..... 1 punct

```
int robiV3(int n, int d[], int u[], int t){
    int nrMinRobinete = 1;
    int timpCrt = 1;
    while (timpCrt <= t){
        int nrCrtRobinete = 0;
        for (int j = 1; j <= n; j++){
                int q = 2 * d[j] + u[j] + 1;
                if (timpCrt % q == 0) //dacă la momentul curent al i-lea robot se află
                        nrCrtRobinete++;                                    // la izvor
        }
        if (nrCrtRobinete > nrMinRobinete)
                nrMinRobinete = nrCrtRobinete;
        timpCrt++;
    }
    return nrMinRobinete - 1;
}
```