CLUJ IULIE 2019

UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Concurs de admitere – 21 iulie 2019

Proba scrisă la Informatică

În atenţia concurenţilor:

1. Se consideră că indexarea şirurilor începe de la 1.
2. 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 acestora se face conform sistemului de punctare parţială din regulamentul concursului.
3. Pentru problemele din Partea B se cer rezolvări complete scrise pe foaia de concurs, fiind evaluate în detaliu conform baremului.
   - **a.** Rezolvarea cerinţelor B1-B4 implică răspunsuri atât în limbaj natural/matematic, cât şi în *pseudocod* sau *Pascal/C/C++*.
   - **b.** Primul criteriu în evaluarea rezolvărilor va fi ***corectitudinea*** algoritmului, iar apoi ***performanţa*** din punct de vedere al *timpului de executare* şi al *spaţiului de memorie utilizat*.
   - **c.** ***Este obligatorie descrierea şi justificarea*** subalgoritmilor **înaintea** rezolvărilor. Se vor scrie, de asemenea, ***comentarii*** pentru a uşura înţelegerea detaliilor tehnice ale soluţiei date, a semnificaţiei identificatorilor, a structurilor de date folosite etc. Neîndeplinirea acestor cerinţe 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).

Partea A (60 puncte)

A.1. Ce se afişează? (6 puncte)

Se consideră programul de mai jos. Care este rezultatul afişat în urma execuţiei programului?

A. `P(a, b) = 253;` `a = 11; b = 11`
B. `P(a, b) = 132;` `a = 11; b = 121;`
C. `P(a, b) = 22;` `a = 11; b = 11;`
D. `P(a, b) = 253;` `a = 11; b = 121;`

Varianta C++

```cpp
#include <iostream>
using namespace std;

int P(int x, int &y){
   y = y * x; x = x + y;
   return x + y;
}
int main(){
   int a = 11; int b = a;
   cout << "P(a, b) = " << P(a, b);
   cout << endl << "; a = " << a;
   cout << "; b = " << b << endl;
   return 0;
}
```

Varianta C

```c
#include <stdio.h>

int P(intx, int *y){
   *y = (*y) * x; x = x + (*y);
   return x + (*y);
}
Int main(){
   int a = 11; int b = a;
   printf("P(a, b) = %d", P(a, &b));
   printf(";\na = %d", a);
   printf("; b = %d\n", b);
   return 0;
}
```

Varianta Pascal

```pascal
function P(x : Integer;
           var y : Integer) : Integer;
begin
   y := y * x; x := x + y;
   P := x + y
end;
var a, b : Integer;
Begin
   a := 11; b := a;
   Write('P(a, b) = ',  P(a, b))
   WriteLn('; ');
   Write('a = ', a);
   WriteLn('; b = ', b);
End.
```

A.2. Evaluare (6 puncte)

Se ştie că v este un vector care conţine n numere naturale mai mici decât 30 000 ($1 \leq n \leq 10000$).
Care dintre următoarele secvenţe de instrucţiuni poate înlocui „ ...” în subalgoritmul `prelucrare(v, n, rez, m)` astfel încât vectorul ***rez***, după executarea structurii repetitive, să reţină valorile care sunt multipli ai lui 5 aflate pe poziţii pare în vectorul ***v***. Lungimea vectorului ***rez*** se reţine în variabila ***m***.

```
Subalgoritm prelucrare(v,n,rez,m)
   m ← 0
   Pentru i = 1, n execută
     ...
   SfPentru
SfSubalgoritm
```

A.
```
Dacă i MOD 2 = 0 ŞI v[i] MOD 5 = 0
   atunci
      m ← m + 1; rez[m] ← v[i]
SfDacă
```

B.
```
Dacă i MOD 2 = 0 atunci
   Dacă v[i] MOD 10 = 5 atunci
      m ← m + 1; rez[m] ← v[i]
   SfDacă
   Dacă v[i] MOD 10 = 0 atunci
      m ← m + 1; rez[m] ← v[i]
   SfDacă
SfDacă
```

C.
```
Dacă v[i] MOD 10 = 0 atunci
   Dacă i MOD 2 = 0 atunci
      m ← m + 1; rez[m] ← v[i]
   altfel
      Dacă v[i] MOD 10 = 5 atunci
         m ← m + 1; rez[m] ← v[i]
      SfDacă
   SfDacă
SfDacă
```

D.
```
Dacă i MOD 2 = 0 ŞI v[i] MOD 10 = 5
        ŞI v[i] MOD 10 = 0 atunci
   m ← m + 1; rez[m] ← v[i]
SfDacă
```

A.3. Ce valori sunt necesare? (6 puncte)

Se consideră subalgoritmul `calcul(n, v)`, unde v este un şir cu n numere întregi (n - număr natural, $1 \leq n \leq 1000$).

```
Subalgoritm calcul(n, v)
  a ← 0; b ← 0
  Pentru i ← 1, n execută
    a ← a + v[i]
  SfPentru
  Pentru i ← 1, n execută
    a ← a - v[i]
    Dacă a = b atunci
      returnează v[i]
    SfDacă
    b ← b + v[i]
  SfPentru
  returnează -1
SfSubalgoritm
```

Precizaţi pentru care valori ale parametrilor subalgoritmul returnează valoarea 0.

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

A.4. Oare ce face? (6 puncte)

```
Subalgoritm ghici(n)
   f ← 0; p ← -1
   Pentru c ← 0, 9 execută
      x ← n; k ← 0
      CâtTimp x > 0 execută
         Dacă x MOD 10 = c atunci
            k ← k + 1
         SfDacă
         x ← x DIV 10
      SfCâtTimp
      Dacă k > f atunci
         p ← c; f ← k
      SfDacă
   SfPentru
   returnează p
SfSubalgoritm
```

Se consideră subalgoritmul `ghici(n)`, unde n este număr natural ($1 \leq n \leq 10000$). Precizaţi efectul subalgoritmului `ghici(n)`:

A. Calculează şi returnează numărul de cifre din reprezentarea zecimală a numărului n.
B. Calculează şi returnează frecvenţa cifrei maxime din reprezentarea zecimală a numărului n.
C. Calculează şi returnează una dintre cifrele cu frecvenţa maximă din reprezentarea zecimală a numărului n.
D. Calculează şi returnează numărul de cifre egale cu c din reprezentarea zecimală a numărului n.

A.5. Generare şir de numere speciale (6 puncte)

Se ştie că într-un şir x cu k elemente $x = (x_1, x_2, x_3, ..., x_k)$ o secvenţă y de lungime p este reprezentată de p elemente ale şirului x care se află pe poziţii consecutive. De ex. $y = (x_3, x_4, x_5, x_6)$ este o secvenţă y de lungime $p = 4$.
Se consideră mulţimea de cifre $M = \{1, 2, ..., n\}$ şi cele $n!$ permutări ale mulţimii M (n – număr natural, $n \leq 9$). Se poate construi cel mai scurt şir ***ss*** format cu cifre din M astfel încât oricare dintre cele $n!$ permutări ale mulţimii M să se regăsească în ***ss*** sub forma unei secvenţe de n cifre.
De exemplu, dacă $n = 3$, avem 6 permutări, iar şirul cerut ***ss*** are 9 cifre şi poate fi, de exemplu următorul: ***ss*** $= (1, 2, 3, 1, 2, 1, 3, 2, 1)$. Din permutarea $perm_1 = (1, 2, 3)$ se pot folosi ultimele două cifre ale sale şi se adaugă o a treia cifră pentru a obţine o altă permutare, adică permutarea $perm_2 = (2, 3, 1)$; apoi se poate aşeza cifra 2 după ultimele două cifre ale $perm_2$ şi se obţine permutarea $perm_3 = (3, 1, 2)$. Dacă după (1, 2) s-ar aşeza cifra 3, s-ar obţine permutarea $perm_1$ care există deja în şirul de cifre generat până la acest moment. Atunci se foloseşte doar ultima cifră din $perm_3$ şi se caută o permutare care începe cu cifra 2 şi încă nu face parte din şir ş. a. m. d. Astfel, în şirul construit există o secvenţă de trei cifre care nu este o permutare (1, 2, 1); restul secvenţelor (adică (1, 2, 3), (2, 3, 1), (3, 1, 2), (2, 1, 3), (1, 3, 2) şi (3, 2, 1)) sunt permutări.
Precizaţi numărul minim de elemente de tip cifră din care se poate construi şirul ***ss*** în cazul mulţimii $M = \{1, 2, 3, 4\}$.

	$perm_1$							
1	2	3	1	2	1	3	2	1
	$perm_2$							

A. 55 **B.** 16 **C.** 33 **D.** 37

A.6. Produs cifre (6 puncte)

```
Subalgoritm cifre(n, d)
  Dacă d = 1 atunci
     Dacă n = 1 atunci
        returnează 0
     altfel
        returnează -1
     SfDacă
  altfel
     Dacă n MOD d = 0 atunci
        val = cifre(n DIV d, d)
        Dacă val < 0 atunci
           returnează -1
        altfel
           returnează val * 10 + d
        SfDacă
     altfel
        returnează cifre(n, d - 1)
     SfDacă
  SfDacă
SfSubalgoritm
```

Se consideră subalgoritmul `cifre(n, d)`, unde n şi d sunt numere naturale ($10 \leq n \leq 100000$, $1 \leq d \leq 9$), care determină şi returnează cel mai mic număr natural format din cifre nenule mai mici sau egale cu d şi cu proprietatea că produsul cifrelor sale este egal cu n. De exemplu, dacă $n = 108$, subalgoritmul returnează 269. Dacă nu există un astfel de număr, subalgoritmul returnează -1. Precizaţi de câte ori se autoapelează subalgoritmul `cifre(n,d)` prin executarea secvenţei de instrucţiuni:

```
Citeşte n
val = cifre(n, 9)
```

A. Dacă $n = 108$, subalgoritmul se autoapelează de 11 ori.
B. Dacă $n = 109$, subalgoritmul se autoapelează de 8 ori.
C. Dacă $n = 13$, subalgoritmul nu se autoapelează niciodată. → 11
D. Dacă $n = 100$, subalgoritmul se autoapelează de 10 ori.

A.7. Prelucrare şir (6 puncte)

Se consideră subalgoritmul `prelucrare(n, x)`, unde x este un şir cu $n - 1$ numere naturale distincte din mulţimea $\{1, 2, ..., n\}$. Precizaţi care este semnificaţia valorii returnate de către subalgoritm.

```
Subalgoritm prelucrare(n, x)
   s ← 0
   Pentru i ← 1, n - 1 execută
      s ← s + x[i]
   SfPentru
   returnează n * (n + 1) / 2 - s
SfSubalgoritm
```

A. Subalgoritmul returnează diferenţa dintre suma primelor n numere naturale nenule şi suma elementelor din şirul x.
B. Subalgoritmul returnează diferenţa dintre suma primelor n numere naturale nenule şi suma elementelor din şir, mai puţin ultimul element din şir.
C. Subalgoritmul returnează acel număr natural din mulţimea $\{1, 2, ..., n\}$ care nu apare în şirul x.
D. Subalgoritmul returnează 0.

?
A.8. Magie (6 puncte)

Un vrăjitor de cifre face o magie prin care un număr natural x ($100 \le x \le 1000000$ și, reprezentat în baza 10, are cel puțin două cifre nenule) se separă în două numere naturale nenule ***stânga*** și ***dreapta***, astfel încât numărul x să se poate scrie prin concatenarea cifrelor numerelor ***stânga*** și ***dreapta***, iar produsul dintre ***stânga*** și ***dreapta*** este maxim. De exemplu, numărul $x = 1092$, prin magie, se separă în numerele ***stânga*** = 10 și ***dreapta*** = 92.

Care dintre subalgoritmii de mai jos aplică magia vrăjitorului asupra unui număr natural x care, reprezentat în baza 10, are cel puțin două cifre nenule ($100 \le x \le 1\,000\,000$), identificând numărul natural z ($1 \le z \le 1\,000\,000$) care reprezintă ***dreapta*** asociată lui x știind că au fost deja definiți subalgoritmii:

- `putere(b, p)` – determină b^p (b la puterea p), b și p fiind numere naturale ($1 \le b \le 20, 1 \le p \le 20$);
- `nrCifre(nr)` – determină numărul cifrelor unui număr natural nr ($1 \le nr \le 1\,000\,000$).

A.
```
Subalgoritm magie(x, z)
   prodMax ← -1
   rez ← 0
   CâtTimp x > 0 execută
      z ← (x MOD 10) * putere(10, nrCifre(z)) + z
      x ← x DIV 10
      Dacă x * z > prodMax atunci
         prodMax ← x * z
         rez ← z
      SfDacă
   SfCâtTimp
   returnează prodMax
SfSubalgoritm
```

B.
```
Subalgoritm magie(x, z)
   t ← 0
   Dacă x > 0 atunci
      y ← (x MOD 10) * putere(10, nrCifre(z)) + z
      t ← x DIV 10
      Dacă x * z < y * t atunci
         returnează magie(y, t)
      altfel
         returnează t
      SfDacă
   altfel
      returnează t
   SfDacă
SfSubalgoritm
```

C.
```
Subalgoritm magie(x, z)
   prodMax ← -1
   rez ← 0
   CâtTimp x > 0 execută
      z ← (x MOD 10) * putere(10, nrCifre(z)) + z
      x ← x DIV 10
      Dacă x * z > prodMax atunci
         prodMax ← x * z
         rez ← z
      SfDacă
   SfCâtTimp
   returnează rez
SfSubalgoritm
```

D.
```
Subalgoritm magie(x, z)
   Dacă x > 0 atunci
      y ← (x MOD 10) * putere(10, nrCifre(z)) + z
      t ← x DIV 10
      Dacă x * z < y * t atunci
         returnează magie(y, t)
      altfel
         returnează z
      SfDacă
   altfel
      returnează z
   SfDacă
SfSubalgoritm
```

A.9. Completați (6 puncte)

Se consideră un șir x ordonat crescător cu n ($3 \le n \le 100$) elemente numere naturale distincte mai mici decât 30 000. Subalgoritmul `apropiat(x, st, dr, val)` determină poziția celui mai mare element din șirul x aflat între pozițiile st și dr ($1 \le st < dr \le n$) și a cărui valoare este mai mică sau egală cu val. În cazul în care nu există un astfel de element, subalgoritmul `apropiat(x, st, dr, val)` returnează 0.

Subalgoritmul `modul(a)` returnează valoarea absolută a numărului întreg a.

Subalgoritmul `calcul(n, x, nr)` determină acea valoare existentă în șirul x care este cea mai apropiată de valoarea lui nr. În cazul a două elemente la fel de apropiate de valoarea lui nr, se va considera elementul mai mare.

Fie $n = 5$, $x = (5, 9, 11, 15, 99)$ și $nr = 12$. Precizați cu ce se pot înlocui punctele de suspensie în cadrul subalgoritmului `apropiat(x, st, dr, val)` pentru ca prin executarea subalgoritmului `calcul(n, x, nr)` să se returneze valoarea 11.

```
Subalgoritm apropiat(x, st, dr, val)
   Dacă val > x[dr] atunci
      returnează dr
   SfDacă
   Dacă val < x[st] atunci
      returnează st - 1
   SfDacă
   mij ← (st + dr) DIV 2
   Dacă ... atunci
      returnează mij - 1
   altfel
      Dacă val < x[mij] atunci
         returnează apropiat(x, st, mij - 1, val)
      altfel
         returnează apropiat(x, mij + 1, dr, val)
      SfDacă
   SfDacă
SfSubalgoritm
```

```
Subalgoritm calcul(n, x, nr)
   i ← apropiat(x, 1, n, nr)
   Dacă i = 0 atunci
      returnează x[i + 1]
   altfel
      Dacă modul(x[i]- nr) < modul(x[i + 1] - nr) atunci
         returnează x[i]
      altfel
         returnează x[i + 1]
      SfDacă
   SfDacă
SfSubalgoritm
```

A. `x[mij - 1] <= val ŞI val < x[mij]`
B. `x[mij - 1] <= val SAU val < x[mij]`
C. `x[mij - 1] < val ŞI val <= x[mij]`
D. `x[mij] <= val ŞI val < x[mij - 1]` → mereu falsa => merge pe else

st dr

A.10. Cavaleri și mincinoși (6 puncte)

Pe o insulă locuiesc cavaleri care spun întotdeauna adevărul și mincinoși care mint întotdeauna. Un vizitator ajuns pe insulă dorește să afle natura unei perechi de localnici cu care s-ar întâlni în plimbările lui pe insulă. Astfel, atunci când el se întâlnește cu doi localnici A și B îi adresează lui A întrebarea „Q_1: Amândoi sunteți cavaleri?”, dar răspunsul primit (R_1) nu îl lămurește în privința naturii fiecăruia din cei doi localnici. De aceea vizitatorul îi adresează tot lui A o nouă întrebare: „Q_2: Sunteți la fel, amândoi cavaleri sau amândoi mincinoși?”; de data aceasta, răspunsul primit (R_2) îl lămurește pe vizitator (adică acum știe ce este fiecare dintre cei doi localnici, cavaler sau mincinos).

Ce a răspuns localnicul A știind că vizitatorul a reușit să identifice în mod exact natura celor doi localnici pe baza scenariului descris anterior?

A. R_1: Da, R_2: Da
B. R_1: Da, R_2: Nu
C. R_1: Nu
D. Variantele B și C sunt corecte.

R_2: ?

?
Partea B (30 puncte)

Banane

În urma unui naufragiu, marinarii de pe o corabie au reușit să se salveze folosind bărcile de salvare, iar fiecare barcă, încărcată cu câte trei marinari, a ajuns pe o altă insulă. Căutând mâncare, marinarii de pe fiecare insulă i au cules b_i banane pe care le-au depozitat în barcă și au decis să le împartă între ei abia a doua zi. În oricare barcă încap cel mult k banane. În timpul nopții, pe o insulă, unul dintre marinari s-a trezit și a împărțit bananele din barcă în trei grămezi, fiecare grămadă având același număr de banane, dar a constatat că i-a mai rămas o banană (neplasată în nici una dintre grămezi) pe care a mâncat-o. Apoi, el a ascuns una dintre grămezi, a pus celelalte două grămezi înapoi în barcă și s-a culcat. Până dimineața fiecare marinar de pe fiecare insulă a efectuat același tip de intervenție secretă (a împărțit bananele în trei grămezi egale, a mâncat banana rămasă, a ascuns una dintre grămezi și a pus înapoi în barcă celelalte două grămezi). Dimineața, pe fiecare insulă, bananele rămase în barcă au fost împărțite în trei grămezi identice nevide și, din nou, a rămas o banană, pe care marinarii au dat-o unei maimuțe. Da, pe fiecare insulă trăia și câte o maimuță!☺

Cerințe:

B.1. Dacă marinarii de pe una din insule au cules înainte de venirea nopții 241 de banane, câte banane au rămas în fiecare dintre cele trei grămezi finale (de pe insula respectivă)? **(2 puncte)**

B.2. Dacă pe una din insule, după ultima împărțire, fiecare din cele trei grămezi conținea 15 banane, iar pe o altă insulă, fiecare din cele trei grămezi conținea 31 banane, câte banane s-au cules în total pe cele două insule? **(2 puncte)**

B.3. Scrieți un subalgoritm care pentru un număr k dat calculează cea mai mare valoare posibilă a lui ***bmax*** care reprezintă numărul de banane culese de marinarii de pe o insulă. Parametrul de intrare al subalgoritmului este k. ***bmax*** este parametru de ieșire. Identificați formula de calcul a valorii maxime a lui ***bmax*** în funcție de valoarea lui k. Explicați raționamentul folosit (***bmax***, k – numere naturale, $1 \le bmax \le k$, $100 \le k \le 10\,000\,000$). **(14 puncte)**

Exemplu: dacă $k = 200$, atunci $bmax = 160$.

B.4. Scrieți un subalgoritm care, pentru o valoare k dată, calculează numărul total maxim de banane culese pe toate insulele (***totalMax***). Se știe că pe fiecare insulă marinarii au cules un număr diferit de banane b_i (k, b_i – numere naturale, $1 \le b_i \le k$, $b_i \ne b_j$, $\forall\ i, j \in \{1, 2, \dots, nr\}$, $100 \le k \le 10\,000\,000$). Parametrii de intrare ai subalgoritmului sunt k și nr – numărul de insule (nr – număr natural, $2 \le nr \le 10$), iar cel de ieșire este ***totalMax***, reprezentând numărul total maxim de banane culese pe toate insulele. Valorile variabilelor k și nr sunt date astfel încât problema are soluție. **(12 puncte)**

Exemplu: dacă $k = 400$, $nr = 3$, atunci pe cele trei insule numărul de banane culese va fi 322, 241 și, respectiv 160. Prin urmare numărul maxim de banane culese pe toate insulele este $totalMax = 322 + 241 + 160 = 723$.

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.