UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

CLUJ 2019 Model 1

Admitere Mate-Info - model
Proba scrisă la Informatică

În atenția concurenților:

1. Se consideră că indexarea şirurilor 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 enunţuri). Obţinerea punctajului aferent problemei este condiţionată de identificarea tuturor variantelor de răspuns corecte şi 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* şi al *spațiului de memorie utilizat*.
   - **c.** ***Este obligatorie descrierea şi justificarea*** (sub) algoritmilor înaintea rezolvărilor. Se vor scrie, de asemenea, ***comentarii*** pentru a uşura 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 şiruri de caractere).

Partea A (60 puncte)

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

Subalgoritmul `generare(n)` prelucrează un număr natural n ($0 < n < 100$).

```
Subalgoritm generare(n):
    nr ← 0
    Pentru i ← 1, 1801 execută
        folosit_i ← fals
    SfPentru
    CâtTimp nu folosit_n execută
        suma ← 0, folosit_n ← adevărat
        CâtTimp (n ≠ 0) execută
            cifra ← n MOD 10, n ← n DIV 10
            suma ← suma + cifra * cifra * cifra
        SfCâtTimp
        n ← suma, nr ← nr + 1
    SfCâtTimp
    returnează nr
SfSubalgoritm
```

Precizați care este efectul acestui subalgoritm.

?
A. calculează, în mod repetat, suma cuburilor cifrelor numărului n până când suma egalează numărul n şi returnează numărul repetărilor efectuate
B. calculează suma cuburilor cifrelor numărului n şi returnează această sumă
C. calculează suma cuburilor cifrelor numărului n, înlocuieşte numărul n cu suma obținută şi returnează această sumă
D. calculează numărul înlocuirilor lui n cu suma cuburilor cifrelor sale până când se obține o valoare calculată anterior sau numărul însuşi şi returnează acest număr

A.2. Ce valori sunt necesare? (6 puncte)

Se consideră subalgoritmul `prelucreaza(v, k)`, unde v este un şir cu k numere naturale ($1 \leq k \leq 1\,000$).

```
Subalgoritm prelucreaza(v, k)
    i ← 1, n ← 0
    CâtTimp i ≤ k și v_i ≠ 0 execută
        y ← v_i, c ← 0
        CâtTimp y > 0 execută
            Dacă y MOD 10 > c atunci
                c ← y MOD 10
            SfDacă
            y ← y DIV 10
        SfCâtTimp
        n ← n * 10 + c
        i ← i + 1
    SfCâtTimp
    returnează n
SfSubalgoritm
```

Precizați pentru care valori ale lui v şi **k** subalgoritmul returnează valoarea 928.

A. v = (194, 121, 782, 0) şi k = 4

B. v = (928) şi k = 1
C. v = (9, 2, 8, 0) şi k = 4
D. v = (8, 2, 9) şi k = 3

A.3. Evaluare logică (6 puncte)

Fie s un şir cu k elemente de tip boolean şi subalgoritmul `evaluare(s, k, i)`, unde k şi i sunt numere naturale ($0 \leq i \leq k \leq 100$).

```
Subalgoritm evaluare(s, k, i)
    Dacă i ≤ k atunci
        Dacă s_i atunci
            returnează s_i
        altfel
            returnează (s_i sau evaluare(s, k, i + 1))
        SfDacă
    altfel
        returnează fals
    SfDacă
SfSubalgoritm
```

Precizați de câte ori se autoapelează subalgoritmul `evaluare(s, k, i)` în urma execuțieiurmătoarei secvențe de instrucțiuni:

```
s ← (fals, fals, fals, fals, fals, fals, adevărat, fals, fals, fals)
k ← 10, i ← 3
evaluare(s, k, i)
```

A. de 3 ori
B. de acelaşi număr de ori ca în următoarea secvență de instrucțiuni

```
s ← (fals, fals, fals, fals, fals, fals, fals, adevărat)
k ← 8, i ← 4
evaluare(s, k, i)
```

C. de 6 ori
D. niciodată

A.4. Reuniune (6 puncte)

Se consideră dat subalgoritmul `aparține(x, a, n)` care verifică dacă un număr natural x aparține mulțimii a cu n elemente; a este un şir cu n elemente şi reprezintă o mulțime de numere naturale ($1 \leq n \leq 200$, $1 \leq x \leq 1000$).

Fie subalgoritmii `reuniune(a, n, b, m, c, p)` şi `calcul(a, n, b, m, c, p)`, descrişi mai jos, unde a, b şi c sunt şiruri care reprezintă mulțimi de numere naturale cu n, m şi respectiv p elemente ($1 \leq n \leq 200$, $1 \leq m \leq 200$, $1 \leq p \leq 400$). Parametrii de intrare sunt a, n, b, m şi p, iar parametrii de ieşire sunt c şi p.

```
1.  Subalgoritm reuniune(a, n, b, m, c, p):
2.      Dacă n = 0 atunci
3.          Pentru i ← 1, m execută
4.              p ← p + 1, c_p ← b_i
5.          SfPentru
6.      altfel
7.          Dacă nu aparține(a_n, b, m) atunci
8.              p ← p + 1, c_p ← a_n
9.          SfDacă
10.         reuniune(a, n - 1, b, m, c, p)
11.     SfDacă
12. SfSubalgoritm
```

```
1.  Subalgoritm calcul(a, n, b, m, c, p):
2.      p ← 0
3.      reuniune(a, n, b, m, c, p)
4.  SfSubalgoritm
```

Precizați care dintre afirmațiile de mai jos sunt întotdeauna adevărate:

A. când mulțimea a conține un singur element, apelul subalgoritmului `calcul(a, n, b, m, c, p)` provoacă apariția unui ciclu infinit
B. când mulțimea a conține 4 elemente, apelul subalgoritmului `calcul(a, n, b, m, c, p)` provoacă executarea instrucțiunii de pe linia 10 a subalgoritmului `reuniune` de 4 ori
C. când mulțimea a conține 5 elemente, apelul subalgoritmului `calcul(a, n, b, m, c, p)` provoacă executarea instrucțiunii de pe linia 2 a subalgoritmului `reuniune` de 5 ori → 6 ori (5+1 cond oprire)
D. când mulțimea a are aceleaşi elemente ca şi mulțimea b, în urma execuției subalgoritmului `calcul(a, n, b, m, c, p)` mulțimea c va avea acelaşi număr de elemente ca şi mulțimea a

A.5. Exponențiere (6 puncte)

Care dintre următorii algoritmi calculează corect valoarea a^b, a şi b fiind două numere naturale ($1 \leq a \leq 11$, $0 \leq b \leq 11$).

A.
```
Subalgoritm expo(a, b):
    rezultat ← 1
```

B.
```
Subalgoritm expo(a, b):
    Dacă b ≠ 0 atunci
```

sub expo(a, b)

rezultat = 1

sub expo(a, b)

dacă b ≠ 0 atunci

```
   CâtTimp b > 0 execută
      Dacă b MOD 2 = 1 atunci
         rezultat ← rezultat * a
      SfDacă
      b ←  b DIV 2
      a ← a * a
   SfCâtTimp
   returnează rezultat
SfSubalgoritm
```

```
      Dacă b MOD 2 = 1 atunci
         returnează expo(a * a, b / 2) * a
      altfel
         returnează expo(a * a, b / 2)
      SfDacă
   altfel
      returnează 1
   SfDacă
SfSubalgoritm
```

C.
```
Subalgoritm expo(a, b):
   rezultat ← 1
   CâtTimp b > 0 execută
      rezultat ← rezultat * a
      b  ← b - 1
   SfCâtTimp
   returnează rezultat
SfSubalgoritm
```

D.
```
Subalgoritm expo(a, b):
   Dacă b = 0 atunci
      returnează 1
   SfDacă
   returnează a * expo(a, b - 1)
SfSubalgoritm
```

A.6. Cel mai mare multiplu (6 puncte)

Care dintre subalgoritmii de mai jos returnează cel mai mare multiplu al numărului natural ***a***, multiplu care este mai mic sau egal cu numărul natural ***b*** (0 < ***a*** < 10 000, 0 < ***b*** < 10 000, ***a*** < ***b***)?

~~A.~~
```
Subalgoritm f(a, b):
   c ← b
   CâtTimp c MOD a = 0 execută
      c ← c – 1
   SfCâtTimp
   returnează c
SfSubalgoritm
```

B.
```
Subalgoritm f(a, b):
   Dacă a < b atunci
      returnează f(2 * a, b)
   altfel
      Dacă a = b atunci
         returnează a
      altfel
         returnează b
      SfDacă
   SfDacă
SfSubalgoritm
```

C.
```
Subalgoritm f(a, b):
   returnează (b DIV a) * a
SfSubalgoritm
```

D.
```
Subalgoritm f(a, b):
   Dacă b MOD a = 0 atunci
      returnează b
   SfDacă
   returnează f(a, b - 1)
SfSubalgoritm
```

?

A.7. Tip de date (6 puncte)

Un tip de date întreg reprezentat pe ***x*** biți (***x*** este număr natural strict pozitiv) va putea reține valori întregi din:

A. $[0, 2^x]$
B. $[0, 2^{x-1}-1]$
C. $[-2^{x-1}, 2^{x-1}-1]$
D. $[-2^x, 2^x-1]$

A.8. Număr apeluri (6 puncte)

Se dă subalgoritmul `f(a, b)`:

```
Subalgoritm f(a, b):
      Dacă a > 1 atunci
            returnează b * f(a - 1, b)
      altfel
            returnează b * f(a + 1, b)
      SfDacă
SfSubalgoritm
```

Precizați de câte ori se auto apelează subalgoritmul `f(a, b)` în urma execuției următoarei secvențe de instrucțiuni:

```
a ← 4
b ← 3
c ← f(a, b)
```

A. de 4 ori
~~B.~~ de 3 ori
C. de o infinitate de ori
D. niciodată

A.9. Șiruri (6 puncte)

Se consideră toate șirurile de lungime ***l*** ∊ {1, 2, 3} formate din litere din mulțimea {*a*, *b*, *c*, *d*, *e*}. Câte dintre aceste șiruri au elementele ordonate strict descrescător și un număr impar de vocale? (*a* și *e* sunt vocale)

A. 14
B. 7
C. 81
D. 78

A.10. Numere pozitive (6 puncte)

Se dă subalgoritmul `numerePozitive(m, a, n, b)`.

```
void numerePozitive(int m,int a[], int &n,
int b[]){
   n = 0;
   for(int i = 1; i <= n; i++){
     if (a[i] > 0){
       n = n + 1;
       b[n] = a[i];
     }
   }
}
```

```
procedure numerePozitive(m:integer;
a:sir; var n:integer; var b:sir)
 begin
   n := 0;
   for i := 1 to n do
     if (a[i] > 0) then
       begin
        n := n + 1;
        b[n] := a[i];
       end;
 end;
```

Care este rezultatul execuției apelului `numerePozitive(k, x, p, y)` pentru ***k*** = 4, șirul ***x***=(-1, 2, -3, 4), ***p*** = -1 și șirul vid ***y*** = ().

A. ***p*** = 3 și ***y*** = (2, 4);
B. ***p*** = 0 și ***y*** = (2, 4);
C. ***p*** = 0 și ***y*** = ();
D. Depinde de valoarea lui ***k***.

Partea B (30 puncte)

B.1. Numere magice (15 puncte)

Se consideră două numere naturale ***p*** și ***q*** (2 ≤ ***p*** ≤ 10, 2 ≤ ***q*** ≤ 10). Un număr natural se numește *magic* dacă mulțimea cifrelor utilizate în scrierea lui în sistemul de numerație având baza ***p*** este identică cu mulțimea cifrelor folosite în scrierea lui în sistemul de numerație având baza ***q***. De exemplu, pentru ***p*** = 9 și ***q*** = 7, $(31)_{10}$ este număr *magic* pentru că $(34)_9 = (43)_7$, iar pentru ***p*** = 3 și ***q*** = 9, $(9)_{10}$ este număr *magic* pentru că $(100)_3 = (10)_9$. Se consideră și subalgoritmul `cifreBază(x, b, c)` pentru determinarea cifrelor numărului ***x*** în baza ***b*** (memorate în șirul ***c***):

```
Subalgoritm cifreBază(x, b, c):
   CâtTimp x > 0 execută
      c[x MOD b] ← 1
      x ← x DIV b
   SfCâtTimp
SfSubalgoritm
```

Cerințe:

a. Scrieți o variantă *recursivă* (fără structuri repetitive) a subalgoritmului `cifreBază(x, b, c)` care are același antet și același efect cu acesta. **(5 puncte)**

?

b. Scrieți modelul matematic al variantei recursive a subalgoritmului `cifreBază(x, b, c)` (dezvoltat la punctul a). **(3 puncte)**

c. Scrieți un subalgoritm care, folosind subalgoritmul `cifreBază(x, b, c)`, pentru două baze ***p*** și ***q*** date determină șirul ***a*** al tuturor numerelor *magice* strict mai mari ca 0 și strict mai mici decât un număr natural ***n*** dat (1 < ***n*** ≤ 10000). Parametrii de intrare ai subalgoritmului sunt ***p*** și ***q*** (cele două baze) și valoarea ***n***. Parametrii de ieșire vor fi șirul ***a*** și lungimea ***k*** a șirului ***a***. **(7 puncte)**

Exemplu: dacă ***p*** = 9, ***q*** = 7 și ***n*** = 500, șirul ***a*** va avea ***k*** = 11 elemente: (1, 2, 3, 4, 5, 6, 31, 99, 198, 248, 297).

Degustare de ciocolată (15 puncte)

O companie de publicitate face reclamă la un nou sortiment de ciocolată şi intenţionează să distribuie mostre de ciocolată la n ($10 \le n \le 10\,000\,000$) copii care sunt aşezaţi într-un cerc. Angajaţii companiei îşi dau seama că distribuirea de mostre tuturor copiilor ar costa foarte mult. În consecinţă, decid să distribuie mostre fiecărui al k-lea ($0 < k < n$) copil din cei n, numărând copiii din k în k (atunci când numărătoarea ajunge la ultimul copil, ea continuă cu primul copil şi aşa mai departe). În numărătoare se vor considera toţi copiii, fie că au primit sau nu ciocolată. Numărătoarea *se opreşte atunci când o ciocolată ar trebui distribuită unui copil care deja a primit.*

a. Cerinţe:Precizaţi proprietatea pe care trebuie să o îndeplinească numerele de ordine ale copiilor care primesc ciocolată. Justificaţi răspunsul. (**3 puncte**)

b. Explicaţi (în limbaj natural şi formule matematice) care este numărul de copii care primesc ciocolată? Justificaţi răspunsul. (**2 puncte**)

c. Scrieţi un subalgoritm care determină numărul copiilor (nr) care nu primesc mostre de ciocolată. Parametrii de intrare sunt numerele naturale n şi k, iar parametrul de ieşire va fi numărul natural nr. (**10 puncte**)

Exemplu 1: dacă $n = 12$ şi $k = 9$, atunci $nr = 8$ (primul, al 2-lea, al 4-lea, al 5-lea, al 7-lea, al 8-lea, al 10-lea, al 11-lea copil nu primesc ciocolată).

Exemplu 2: dacă $n = 15$ şi $k = 7$, atunci $nr = 0$ (toţi copiii primesc ciocolată).

Notă:

1. Toate subiectele sunt obligatorii.
2. Rezolvările trebuie scrise detaliat pe foile de examen (ciornele nu se iau în considerare).
3. Se acordă 10 puncte din oficiu.
4. Timpul efectiv de lucru este de 3 ore.