

Concurs admitere septembrie 2018
Proba scrisă la Informatică

În atenția concurenților:

- Se consideră că indexarea tuturor șirurilor începe de la 1.
- 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.
- Pentru problemele din Partea B se cer rezolvări complete pe foaia de concurs.
 - Rezolvările se vor scrie în pseudocod sau într-un limbaj de programare (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 (sub) algoritmilor înaintea rezolvărilor. Se vor scrie, de asemenea, **comentarii** pentru a ușura înțelegerea detaliilor tehnice ale soluției date, a semnificației identificărilor, 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 (30 puncte)

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

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

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

Precizați care este efectul acestui subalgoritm.

- calculează, în mod repetat, suma cuburilor cifrelor numărului n până când suma egalează numărul n și returnează numărul repetărilor efectuate
- calculează suma cuburilor cifrelor numărului n și returnează această sumă
- calculează suma cuburilor cifrelor numărului n , înlocuiește numărul n cu suma obținută și returnează această sumă
- calculează, în mod repetat, suma cuburilor cifrelor numărului n până când o sumă se obține a doua oară și returnează numărul repetărilor efectuate
- calculează numărul înlocuirilor lui n cu suma cuburilor cifrelor sale până când se obține o valoare calculată anterior sau numărul însuși și returnează acest număr

A.2. Ce valori sunt necesare? (5 puncte)

Se consideră subalgoritmul `prelucreaza(v, k)`, unde v este un șir cu k numere naturale ($1 \leq k \leq 1000$).

```
Subalgoritm prelucreaza(v, k)
    i ← 1, n ← 0
    Câtîmp i ≤ k și  $v_i \neq 0$  execută
        y ←  $v_i$ , c ← 0
        Câtîmp y > 0 execută
            Dacă y MOD 10 > c atunci
                c ← y MOD 10
            SfDacă
            y ← y DIV 10
        SfCâtîmp
        n ← n * 10 + c
        i ← i + 1
    SfCâtîmp
    returnează n
SfSubalgoritm
```

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

- $v = (124, 121, 782, 0)$ și $k = 4$
- $v = (928)$ și $k = 1$
- $v = (9, 2, 8, 0)$ și $k = 4$
- $v = (8, 2, 9)$ și $k = 3$
- $v = (912, 0, 120, 8, 0)$ și $k = 5$

A.3. Evaluare logică (5 puncte)

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

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

Precizați de câte ori se autoapelează subalgoritmul `evaluare(s, k, i)` în următoarea secvență de instrucțiuni:

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

- de 3 ori
- de același 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)
```

- de 6 ori
- niciodată
- de o infinitate de ori

A.4. Reuniune (5 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 șir cu n elemente și reprezintă o mulțime de numere naturale ($1 \leq n \leq 200, 1 \leq x \leq 1000$).

Fie subalgoritmul `reuniune(a, n, b, m, c, p)` și `calcul(a, n, b, m, c, p)`, descriși mai jos, unde a, b și c sunt șiruri care reprezintă mulțimi de numere naturale cu n, m și respectiv p elemente ($1 \leq n \leq 200, 1 \leq m \leq 200, 1 \leq p \leq 400$). Parametri de intrare sunt a, n, b, m și p , iar parametri de ieșire sunt c și p .

| | |
|--|---|
| <pre>1. Subalgoritm reuniune(a, n, b, m, c, p): 2. Dacă n = 0 atunci 3. Pentru i ← 1, m execută 4. p ← p + 1 5. c_p ← b_i 6. SfPentru 7. altfel 8. Dacă nu aparține(a, b, m) atunci 9. p ← p + 1 10. c_p ← a 11. SfDacă 12. reuniune(a, n - 1, b, m, c, p) 13. SfDacă 14. SfSubalgoritm</pre> | <pre>1. Subalgoritm calcul(a, n, b, m, c, p): 2. p ← 0 3. reuniune(a, n, b, m, c, p) 4. SfSubalgoritm</pre> |
|--|---|

Precizați care dintre afirmațiile de mai jos sunt întotdeauna adevărate:

- când mulțimea a conține un singur element, apelul subalgoritmului `calcul(a, n, b, m, c, p)` provoacă apariția unui ciclu infinit
- când mulțimea a conține 4 elemente, apelul subalgoritmului `calcul(a, n, b, m, c, p)` provoacă executarea instrucțiunii de pe linia 12 a subalgoritmului `reuniune` de 4 ori
- 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
- când mulțimea a are același număr de elemente ca și mulțimea b , în urma execuției subalgoritmului `calcul(a, n, b, m, c, p)` mulțimea c va avea același număr de elemente ca și mulțimea a
- când mulțimea a are aceleași elemente ca și mulțimea b , în urma execuției subalgoritmului `calcul(a, n, b, m, c, p)` mulțimea c va avea același număr de elemente ca și mulțimea a

A.5. Exponențiere (5 puncte)

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

| | |
|---|---|
| <p>A. Subalgoritm expo(a, b): rezultat ← 1 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</p> | <p>B. Subalgoritm expo(a, b): Dacă b = 0 atunci 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</p> |
| <p>C. Subalgoritm expo(a, b): rezultat ← 1 CâtTimp b > 0 execută rezultat ← rezultat * a b ← b - 1 SfCâtTimp returnează rezultat SfSubalgoritm</p> | <p>D. Subalgoritm expo(a, b): Dacă b = 0 atunci returnează 1 SfDacă aux ← expo(a, b DIV 2) Dacă b MOD 2 = 0 atunci returnează aux * aux altfel returnează a * aux * aux SfDacă SfSubalgoritm</p> |
| <p>E. Subalgoritm expo(a, b): Dacă b = 0 atunci returnează 1 SfDacă returnează a * expo(a, b - 1) SfSubalgoritm</p> | |

A.6. Cel mai mare multiplu (5 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 ($0 < a < 10\,000, 0 < b < 10\,000, a < b$)?

| | |
|--|--|
| <p>A. Subalgoritm f(a, b): c ← b CâtTimp c MOD a = 0 execută c ← c - 1 SfCâtTimp returnează c SfSubalgoritm</p> | <p>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 SfSubalgoritm</p> |
| <p>C. Subalgoritm f(a, b): returnează (b DIV a) * a SfSubalgoritm</p> | |
| <p>D. Subalgoritm f(a, b): Dacă b MOD a = 0 atunci returnează b SfDacă returnează f(a, b - 1) SfSubalgoritm</p> | <p>E. Subalgoritm f(a, b): c ← a CâtTimp c < b execută c ← c + a SfCâtTimp Dacă c = b atunci returnează c altfel returnează c - a SfDacă SfSubalgoritm</p> |

Partea B (60 puncte)

B.1. Evaluare polinom (10 puncte)

Se consideră subalgoritmul evaluare(n , coef, x), unde *coef* este un vector cu $n + 1$ elemente numere reale din intervalul $[-100, 100]$ reprezentând coeficienții unui polinom de grad n , $P(x) = \text{coef}_1 * x^n + \text{coef}_2 * x^{n-1} + \dots + \text{coef}_n * x + \text{coef}_{n+1}$, dați în ordinea descrescătoare a puterilor lui x (n este număr natural, $1 \leq n \leq 10$). Subalgoritmul determină valoarea polinomului într-un punct dat x (x număr real din intervalul $[-10, 10]$).

```
Subalgoritm evaluare(n, coef, x):
    val ← 0.0
    Pentru i ← 1, n + 1 execută
        val ← val * x + coef[i]
    SfPentru
    returnează val
SfSubalgoritm
```

Scrieți o variantă *recursivă* (care nu conține structuri repetitive) a subalgoritmului evaluare(n , coef, x) care are același antet și același efect cu acesta.

B.2. Intersecție (25 puncte)

Se consideră două șiruri, fiecare conținând numere întregi *distincte*, cuprinse între -30 000 și 30 000. Șirul a are n ($0 < n \leq 10\,000$) elemente, iar șirul b are m ($0 < m \leq 10\,000$) elemente și este *ordonat crescător*.

Scrieți un subalgoritm care determină șirul c , având k ($0 \leq k \leq 10\,000$) elemente, format din toate elementele *comune* ale celor două șiruri, luate o singură dată în orice ordine. Parametrii de intrare sunt cele două șiruri (a și b) și lungimile lor (n și m). Parametrii de ieșire vor fi șirul c și lungimea k a șirului. Dacă nu există elemente comune, k va fi 0.

Exemplu: dacă $n = 4$, $a = (5, -7, -2, 3)$, $m = 5$ și $b = (-2, 3, 5, 7, 8)$, șirul c are $k = 3$ elemente și este $c = (5, -2, 3)$.

B.3. Secvență de numere fără frați (25 puncte)

Se consideră un șir x , având n ($0 < n \leq 10\,000$) elemente numere naturale *distincte* nenule mai mici decât 30 000.

Două numere se numesc *frați* dacă *sunt distincte* și dacă *au cel puțin două cifre distincte comune*. De exemplu, 5867 și 17526 sunt *frați*, dar 5867 și 152 nu sunt *frați*. De asemenea, 131 și 114 nu sunt frați.

Cerințe:

- Scrieți un subalgoritm care verifică dacă un număr natural a este frate cu un număr natural b ($0 < a \leq 30\,000, 0 < b \leq 30\,000$). Parametrii de intrare sunt cele două numere a și b . Parametrul de ieșire va fi *esteFrate* și va avea valoarea *adevărat* dacă a este frate cu b și *fals*, altfel. (11 puncte)
- Scrieți un subalgoritm care determină cea mai lungă subsecvență a șirului x , formată din elemente care *nu au niciun frate* în șirul x . O subsecvență a unui șir este formată din elemente ale șirului aflate pe poziții consecutive. Parametrii de intrare sunt șirul x și lungimea lui n . Parametrii de ieșire vor fi poziția de început a subsecvenței *start* și lungimea acesteia k . Dacă există mai multe subsecvențe de aceeași lungime maximă, se va considera ultima dintre ele. Dacă nu există nicio astfel de subsecvență, *start* va fi -1 și k va fi 0. (14 puncte)

Exemplu: Fie $n = 11$ și $x = (12345, 9, 100, 567, 5678, 345, 123, 8989, 222, 11, 78)$. Numerele fără frați din șirul x sunt: 9, 100, 8989, 222, 11, iar subsecvența căutată este (8989, 222, 11), deci *start* = 8 și $k = 3$.

Notă:

- Toate subiectele sunt obligatorii.
- Ciornele nu se iau în considerare.
- Se acordă 10 puncte din oficiu.
- Timpul efectiv de lucru este de 3 ore.

OFICIU 10 puncte

| | |
|---|------------------|
| Partea A | 30 puncte |
| A. 1. Oare ce face? Răspunsul E | 5 puncte |
| A. 2. Ce valori sunt necesare? Răspunsurile A, C | 5 puncte |
| A. 3. Evaluare logică. Răspunsul B | 5 puncte |
| A. 4. Reuniune. Răspunsurile B, E | 5 puncte |
| A. 5. Exponențiere. Răspunsurile A, B, C, D, E | 5 puncte |
| A. 6. Cel mai mare multiplu. Răspunsurile C, D, E | 5 puncte |

Partea B **60 puncte****B. 1. Evaluare polinom** **10 puncte**

- respectarea parametrilor de intrare și ieșire 2 puncte
- condiția de oprire din recursivitate 2 puncte
- autoapel 2 puncte
- valoarea returnată la oprirea recursivității 2 puncte
- valoarea returnată la continuarea recursivității 2 puncte

B. 2. Intersecție **25 puncte**

- respectarea parametrilor de intrare și ieșire 2 puncte
 - variante:
 - folosirea căutării binare 23 puncte
 - fără căutare binară maxim 18 puncte

B. 3. Secvență de numere fără frați **25 puncte**

- respectarea parametrilor de intrare și ieșire 2 puncte
- proprietatea de frate 10 puncte
- determinarea unei secvențe 9 puncte
- determinarea celei mai lungi secvențe 4 puncte