UNIVERSITATEA DE VEST DIN TIMIŞOARA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Examen de admitere
Informatică, 10 Iulie 2022

Notă: Subprogramele/funcțiile solicitate pot fi scrise în limbajele de programare C/C++/ Pascal/Python. *Proba de examen durează 3 ore.*

1. (1p) Justificaţi pe scurt răspunsurile selectate.

   (a) (0.25p) Se consideră suma $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \ldots$. Care este forma generală a celui de al i-lea termen? (în ipoteza că termenii sunt numerotaţi începând cu $i = 1$).

   (i) $\frac{x^i}{i!}$ (ii) $\frac{x^{i-1}}{(i-1)!}$ (iii) $\frac{x^i}{(i-1)!}$ (iv) $\frac{x^{i-1}}{i!}$

   (b) (0.25p) Se consideră două mulţimi $A, B \subset \{1, 2, \ldots, n\}$ reprezentate prin tablouri cu câte n elemente definite prin:

   $$a_i = \begin{cases} 1 & \text{dacă } i \in A \\ 0 & \text{dacă } i \notin A \end{cases} \qquad b_i = \begin{cases} 1 & \text{dacă } i \in B \\ 0 & \text{dacă } i \notin B \end{cases}, \qquad i = \overline{1, n}$$

   Care dintre expresiile următoare corespund numărului de elemente din $A \cap B$?

   (i) $\sum_{i=1}^{n} a_i b_i$ (ii) $\sum_{i=1}^{n} (a_i + b_i - a_i b_i)$ (iii) $\sum_{i=1}^{n} \min\{a_i, b_i\}$ (iv) $\sum_{i=1}^{n} \max\{a_i, b_i\}$

   (c) (0.25p) Se consideră intervalele $[a_1, b_1] \subset \mathbb{R}$ şi $[a_2, b_2] \subset \mathbb{R}$. In ipoteza că $a_1 \leq a_2$, care dintre următoarele expresii este adevărată în oricare dintre cazurile în care intersecţia dintre intervale este nevidă?

   (i) $a_2 \leq b_1$ (ii) $b_2 \leq b_1$ (iii) $b_2 \geq a_1$ (iv) $a_1 \leq a_2$ şi $b_2 \leq b_1$

   (d) (0.25p) Se consideră o funcţie $C : \{0,1\}^8 \to \{0,1\}^8$ care transformă un şir de 8 biţi după o anumită regulă. Ştiind că $C(0,1,1,0,0,1,0,0) = (1,0,0,1,1,1,0,0)$, $C(0,1,0,1,1,0,1,1) = (1,0,1,0,0,1,0,1)$ şi $C(0,0,0,0,0,0,1,0) = (1,1,1,1,1,1,1,0)$ descrieţi pe scurt regula de transformare şi specificaţi care este valoarea lui $C(1,1,1,1,1,1,1,1)$.

2. (1p) Se consideră o secvenţă a_1, a_2, …, a_n de numere întregi şi relaţia de recurenţă:

   $$f(i,j) = \begin{cases} 1 & \text{dacă } i \geq j \\ 0 & \text{dacă } i < j \text{ şi } a_i \neq a_j \\ f(i+1, j-1) & \text{dacă } i < j \text{ şi } a_i = a_j \end{cases}$$

   cu $i, j \in \{1, \ldots, n\}$.

   (a) (0.5p) Pentru care dintre secvenţele următoare, valoarea funcţiei $f(1, n)$ este 1? Descrieţi proprietatea comună a acestor secvenţe.

   (i) 1, 2, 2, 1 (ii) 1, 2, 4, 2, 1 (iii) 2 (iv) 1, 2, 1, 2 (v) 1, 1, 1, 1 (vi) 1, 1, 1, 1, 1

   (b) (0.5p) Care este numărul maxim de comparaţii (între elemente ale secvenţei) care se pot executa la evaluarea funcţiei $f(1, n)$ pentru un n dat şi o secvenţă a arbitrară?

3. (1p) Se consideră o matrice A cu m linii și n coloane ($4 \leq m \leq 50$, $4 \leq n \leq 50$), având elemente numere reale. Se pune problema construirii matricii B cu aceleaşi dimensiuni, ale cărei elemente satisfac relaţia:

   $$B_{ij} = \sum_{q=1}^{i} \sum_{r=1}^{j} A_{qr}, \qquad i = \overline{1, m}, j = \overline{1, n}.$$

   (a) (0.25p) Propuneţi o strategie de construire a matricii B astfel încât să fie efectuate cât mai puţine operaţii de adunare.

   (b) (0.5p) Folosind strategia propusă la punctul anterior scrieţi un subprogram care completează elementele matricii B. Se consideră că matricile A şi B sunt stocate în variabile globale.

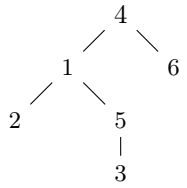
(c) (0.25p) Se consideră indicii $1 \leq i_1 < i_2 \leq m$ şi $1 \leq j_1 < j_2 \leq n$. Propuneţi o modalitate de calcul a sumei elementelor din submatricea $A[i_1 + 1..i_2, j_1 + 1..j_2]$ care să utilizeze doar elementele cu indicii (i_1, j_1), (i_1, j_2), (i_2, j_1), (i_2, j_2) din matricea B.

4. (1.5p) Se consideră un şir constituit din minim 7 şi maxim 15 caractere (caracterele sunt doar din mulţimea {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '.'} iar şirul nu poate conţine simboluri '.' consecutive) şi se pune problema verificării faptului că şirul respectă cerinţele de specificare ale unei adrese IP, adică are o structură de forma ⟨nr1⟩.⟨nr2⟩.⟨nr3⟩.⟨nr4⟩ unde nr1, nr2, nr3 şi nr4 sunt valori naturale din mulţimea $\{0, \ldots, 255\}$, iar nr1 nu poate fi 0. Un exemplu de adresă IP este 10.136.18.211

   (a) (1p) Scrieţi un subprogram care primeşte ca parametru de intrare un şir de caractere care satisface cerinţele din enunţ şi completează într-un parametru de ieşire v (de tip tablou) valorile naturale corespunzătoare subsecvenţelor de cifre separate de simbolul '.'. In cazul şirului '10.136.18.211' se obţine tabloul cu elementele 10, 136, 18 şi 211.

   (b) (0.5p) Scrieţi un subprogram care primeşte ca parametru de intrare un şir de caractere care satisface cerinţele din enunţ şi care returnează 1 dacă şirul este o adresă IP validă şi 0 în caz contrar. Se vor utiliza rezultatele obţinute la punctul anterior.

5. (1.5p) Se consideră dat un arbore cu n noduri numerotate de la 1 la n, reprezentat ca un tablou T cu $n + 1$ elemente ($n \leq 100$), astfel încât $T[i]$ este părintele nodului i. Rădăcina arborelui este singurul nod r care nu are părinte, iar în acest caz $T[r] = 0$. Elementul $T[0]$ al tabloului nu este folosit. De exemplu, tabloul T cu $T[1] = 4$, $T[2] = 1$, $T[3] = 5$, $T[4] = 0$, $T[5] = 1$ şi $T[6] = 4$ reprezintă arborele



   (a) (0.5p) Nivelul unui nod în arbore este numărul de muchii care trebuie parcurse pentru a se ajunge de la nodul respectiv la nodul rădăcină. De exemplu, nodurile 1 şi 6 sunt pe nivelul 1, nodurile 2 şi 5 pe nivelul 2, iar nodul 3 pe nivelul 3. Scrieţi un subprogram care primeşte ca parametru tabloul T şi indicele unui nod şi returnează nivelul din arbore pe care se află nodul.

   (b) (1p) Primul ascendent comun a două noduri distincte este nodul cu cel mai mare nivel care se află în intersecţia dintre ramurile care unesc cele două noduri cu nodul rădăcină. De exemplu primul ascendent comun pentru nodurile 2 şi 3 este nodul 1, iar primul ascendent comun pentru nodurile 2 şi 6 este nodul 4. Descrieţi în limbaj natural modul în care poate fi calculat primul ascendent comun a două noduri şi scrieţi un subprogram care primeşte ca parametri de intrare tabloul T, indicii a două noduri nod1 şi nod2 şi returnează indicele primului ascendent comun.

6. (1.5p) O *pauză* într-un şir de biţi este o secvenţă de unu sau mai mulţi biţi 0 care separă doi biţi 1. De exemplu, şirul de biţi 001<u>000</u>111<u>00</u>10 are două pauze: 000 şi 00.

   (a) (0.5p) Câte şiruri de 7 biţi conţin o singură pauză, iar lungimea pauzei este 3?

   (i) 1 (ii) 5 (iii) 10 (iv) 15 (v) 20 (vi) 25

   (b) (1p) Scrieţi un subprogram care primeşte ca parametru de intrare o secvenţă de valori binare şi returnează numărul de pauze din secvenţă.

7. (1.5p) La secţia de împachetare a produselor dintr-o fabrică, lucrează n muncitori ($n \leq 20$). Fiecare muncitor împachetează acelaşi tip de produs, şi pentru fiecare muncitor se cunoaşte timpul necesar, exprimat în minute, pentru împachetarea unui obiect (timpii se consideră stocaţi într-un tablou T cu n valori naturale nenule). Se pune problema determinării numărului minim de minute în care cei n muncitori împachetează împreună cel puţin M obiecte. De exemplu pentru cazul în care $n = 6$ (sunt 6 muncitori), $M = 60$ (60 de obiecte), iar timpii de împachetare sunt $T = (4, 7, 3, 6, 7, 1)$ durata minimă este 30 minute.

   (a) (0.5p) Descrieţi în limbaj natural ideea de rezolvare a problemei.

   (b) (1p) Scrieţi un subprogram care primeşte ca parametri: numărul de muncitori, numărul de obiecte, tabloul cu duratele de împachetare şi returnează durata minimă.

BAREM

1 (a) Răspunsul corect este (ii) $\frac{x^{i-1}}{(i-1)!}$ 0.25p

(b) Numărul de elemente din $A \cap B$ este numărul de indici i pentru care $a_i = 1$ şi $b_i = 1$, ceea ce este echivalent cu $a_i b_i = 1$ sau cu $\min\{a_i, b_i\} = 1$, deci răspunsurile corecte sunt (i) şi (iii) 0.25p
(c) In ipoteza că $a_1 \leq a_2$, pentru ca intersecţia să fie nevidă este suficient şi necesar ca (i) $a_2 \leq b_1$ 0.25p
(d) Transformarea presupune parcurgerea biţilor de la dreapta la stânga până la întâlnirea primului bit egal cu 1. Biţii parcurşi, inclusiv primul bit egal cu 1 sunt lăsaţi nemodificaţi. Toţi biţii parcurşi în continuare sunt complementaţi (0 este transformat în 1 şi 1 este transformat în 0). $C(1,1,1,1,1,1,1,1) = (0,0,0,0,0,0,0,1)$ 0.25p

2 (a) Funcţia returnează 1 pentru orice secvenţă care are proprietatea că $a_i = a_{n-i+1}$, $i = \overline{1,n}$ (e palindrom) 0.25p
Răspunsuri corecte: (i), (ii), (iii), (v), (vi) 0.25p

(b) Pentru o secvenţă cu n elemente, numărul maxim de comparaţii între elementele secvenţei se obţine când secvenţa este de tip palindrom, ceea ce înseamnă că numărul maxim de comparaţii este $[n/2]$ 0.5p

3 (a) Pentru a reduce numărul de operaţii trebuie evitată recalcularea sumelor parţiale observând că $B_{ij} = B_{i-1,j} + B_{i,j-1} - B_{i-1,j-1} + A_{ij}$ pentru $1 < i \leq m$, $1 < j \leq n$. Pornind de la elementul $B_{11} = A_{11}$ se completează elementele primei linii conform regulii $B_{1j} = B_{1,j-1} + A_{ij}$, ale primei coloane conform regulii $B_{i1} = B_{i-1,1} + A_{ij}$ după care se completează celelalte elemente linie după linie folosind regula de mai sus. 0.25p

(b) O variantă de implementare bazată pe ideea de mai sus este descrisă mai jos. 0.5p

```
void construireB(int m, int n)
{int i,j;
 B[0][0] = A[0][0];
 for (i=1;i<m;i++)   // (m-1) operatii de adunare
   B[i][0] = B[i-1][0]+A[i][0];
 for (j=1;j<n;j++)   // (n-1) operatii de adunare
   B[0][j] = B[0][j-1]+A[0][j];
 for (i=1;i<m;i++)   // 2(m-1)(n-1) operatii de adunare si (m-1)(n-1) operatii de scadere
   for (j=1;j<n;j++)
     B[i][j] = B[i-1][j] + B[i][j-1] - B[i-1][j-1] + A[i][j];
}
```

Pentru variantele de rezolvare pentru care numărul de adunări este de ordinul m^2n^2 se acordă 0.25p.

(c) Suma elementelor din submatricea $A[i_1+1..j_1, i_2+1..j_2]$ se obţine efectuând calculul $B[i_2][j_2] - B[i_2][j_1] - B[i_1][j_2] + B[i_1][j_1]$ 0.5p

4 (a) Se parcurge şirul şi se converteşte în baza 10 fiecare secvenţă de cifre cuprinse între separatoarele de tip '.' sau aflată la începutul sau sfârşitul şirului. O variantă de implementare a subprogramului este: 1p

```
void construire(char sir[], int& n, int v[])
{int i,lg;
 int nr=0;
 lg = strlen(sir);
 i = 0; n = -1;
 while (i<lg)
  {if (sir[i]=='.') {n = n+1; v[n] = nr; nr = 0;}
   else nr = 10*nr+(sir[i]-'0');
   i++;
  }
  if(sir[i-1]!='.') {n = n+1; v[n] = nr;}
}
```

(b) Se verifică faptul că tabloul v construit la pasul anterior conţine 4 valori numerice din $\{0, \ldots, 255\}$, iar prima valoare este diferită de 0. O variantă de implementare este descrisă mai jos 0.5p

```
int verific(char sir[])
{int n, v[15];
 construire(sir, n, v);
```

```
 if (n!=3) return 0;
 if (v[0]==0) return 0;
 for (int i=0;i<n;i++)
  if (v[i]<0 || v[i]>255) return 0;
 return 1;
}
```

5 (a) Este suficient să se contorizeze numărul de noduri părinte întâlnite până se ajunge la nodul rădăcină. O variantă de implementare este: 0.5p

```
int nivel(int T[], int nod)
{int k=0;
 while (T[nod]!=0) {k++; nod=T[nod];}
 return k;
}
```

(b) O variantă de rezolvare este să se marcheze într-un tablou binar indicii nodurilor vizitate pornind de la nodul nod1 şi parcurgând ramura prin legătura către părinte a fiecărui nod vizitat, după care să se pornească de la nod2 şi să se oprească parcurgerea la primul nod deja vizitat (acesta va fi ascendentul comun căutat).

O variantă de implementare este:

```
#define NMAX 101
int ascendent(int n, int T[], int nod1, int nod2)
{int vizitat[NMAX];
 int nod;
 for (int i=1;i<=n;i++) vizitat[i]=0;
 nod = nod1; vizitat[nod] = 1;
 while (T[nod]!=0) {nod=T[nod]; vizitat[nod]=1;}
 nod = nod2;
 while (vizitat[nod]==0) nod=T[nod];
 return nod;
}
```

6 (a) Orice şir de 7 biţi care are doar pauza 000 este de forma $0^a1^b100011^c0^d$ cu $a, b, c, d \in \mathbb{N}$ şi $a+b+c+d = 2$. Numărul de astfel de şiruri de biţi coincide cu numărul de soluţii $a, b, c, d \in \mathbb{N}$ cu $a + b + c + d = 2$. Sunt 10 soluţii. 0.5p

(b) Este suficient să se contorizeze numărul de perechi de forma 01. In cazul în care secvenţa începe cu 0 se va ignora prima pereche întâlnită. O variantă de rezolvare (cu secvenţa reprezentată ca şir de caractere este 1p

```
int pauze(char s[])
{int i=1, n=strlen(s);
 int nr=0;
 while (i<n)
  {if (s[i-1]=='0' && s[i]=='1')  nr = nr+1;
   i = i+1;
  }
 if (s[0]==0) nr = nr-1;
 return nr;
}
```

7 (a) Se simulează trecerea timpului printr-o prelucrare iterativă în cadrul căreia se incrementează o variabilă care contorizează numărul de minute. De fiecare dată când se ajunge la un număr de minute care e multiplu al duratei de împachetare corespunzătoare oricăruia dintre muncitori se consideră că a mai fost împachetat un obiect. 0.5p

(b) O variantă de rezolvare este 1p

```
int durata(int n, int M, int T[])
{int timp=0;
 while (M>0)
   {timp = timp+1;
    for (int i=0;i<n;i++)
```

```
        if (timp%T[i]==0) M = M-1; // muncitorul i a impachetat inca un obiect
      }
     return timp;
   }
```

Notă. Orice variantă corectă de rezolvare este punctată corespunzător.