UNIVERSITATEA DE VEST DIN TIMIŞOARA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

CULEGERE DE PROBLEME PENTRU ADMITERE

Informatică

Material elaborat în cadrul proiectului CNFIS-FDI-2021-0471 „UVT – Acces şi echitate în învățământul superior”

TIMIŞOARA
2021

Cuprins

1 Subiecte propuse în cadrul sesiunilor de pregătire pentru bacalaureat/admitere la facultate **3**

1.1 Algoritmi de prelucrare a datelor simple 3

1.2 Descompunerea unei probleme în subprobleme 7

1.3 Prelucrarea tablourilor unidimensionale 9

1.4 Prelucrarea tablourilor bidimensionale 16

1.5 Algoritmi recursivi 20

1.6 Algoritmi de sortare. Algoritmi de căutare 21

1.7 Prelucrări asupra şirurilor de caractere 30

1.8 Structuri de date pentru grafuri 40

2 Subiecte date la Concursul FMI **44**

2.1 Ediția a III-a (2017) 44

2.2 Ediția a IV-a (2018) 47

2.3 Ediția a V-a (2019) 51

2.4 Ediția a VI-a (2021) 57

3 Subiecte date la examenul de admitere la facultate **68**

3.1 Sesiunea Iulie 2016 68

3.2 Sesiunea Iulie 2017 72

3.3 Sesiunea Iulie 2018 77

3.4 Sesiunea Iulie 2019 81

Capitolul 1

Subiecte propuse în cadrul sesiunilor de pregătire pentru bacalaureat/admitere la facultate

1.1 Algoritmi de prelucrare a datelor simple

Tematică

- Prelucrări asupra cifrelor (extragerea şi analiza cifrelor unui număr natural)
- Proprietăţi de divizibilitate (verificare divizibilitate, determinarea celui mai mare divizor comun, verificare şi generare numere prime, descompunere în factori primi)
- Calcule de sume şi produse
- Generarea de şiruri date prin relaţii de recurenţă

Sumar teoretic

Rezolvarea algoritmică a problemelor presupune parcurgerea următoarelor etape:

- Identificarea datelor de intrare, a celor de ieșire și a proprietăților acestora.
- Stabilirea tipului variabilelor corespunzătoare datelor de intrare, de ieșire și a celor de lucru.
- Identificarea metodei de rezolvare.
- Descrierea în pseudocod a algoritmului.
- Implementarea algoritmului într-un limbaj de programare.
- Verificarea algoritmului/ testarea programului.

Orice algoritm este constituit dintr-o succesiune de prelucrări aplicate asupra unui set de date. Datele cu care se operează în cadrul unui algoritm pot fi *simple* (nestructurate) sau *structurate*. Din perspectiva modului în care sunt utilizate, datele pot fi constante (valoarea lor nu se modifică pe parcursul execuției algoritmului/ programului) sau variabile (valorile pot fi modificate prin atribuire sau prin operații de citire).

- Datele simple pot fi: (i) numerice (cu valori întregi sau reale); (ii) logice (cu două valori posibile: adevărat sau fals); (iii) simboluri (caractere).
- Datele structurate pot fi *omogene* (toate componentele au același tip) sau *neomogene* (componentele pot avea tipuri diferite). Cele mai frecvent utilizate structuri omogene sunt tablourile (unidimensionale - vectori, bidimensionale - matrici, multidimensionale). Elementele unui tablou sunt referite folosind indici specificați între paranteze drepte. Datele neomogene sunt caracterizate prin faptul că sunt constituite din câmpuri care pot avea tipuri diferite și sunt cunoscute cu denumirea de structuri sau înregistrări (`struct` în C, `record` în Pascal). Câmpurile unei structuri sunt referite prin calificare (se specifică numele structurii și numele câmpului separate prin punct).

Prelucrările specifice unui algoritm sunt:

- *Atribuire:* unei variabile i se asignează valoarea obținută prin evaluarea unei expresii; în felul acesta se stochează în variabile rezultate ale calculelor. Operatorul de atribuire depinde de limbajul de programare utilizat (= în C, := în Pascal). Expresiile permit descrierea unor calcule și sunt constituite din operanzi (pot fi variabile sau constante) și operatori (artitmetici, relaționali sau logici).
- *Operații de intrare/ieșire:* permit preluarea de valori corespunzătoare datelor de intrare (operații de citire) și afișarea valorilor corespunzătoare datelor de ieșire (operații de scriere).
- *Decizii (prelucrări condiționale):* permit ramificarea prelucrării în funcție de îndeplinirea unei condiții (specificată printr-o expresie relațională sau logică); varianta cea mai frecvent utilizată de prelucrare condițională este cea specificată prin instrucțiune de tip `dacă (<condiție) atunci <prelucrare 1> altfel <prelucrare 1>`.
- *Cicluri (prelucrări repetitive):* permit execuția repetată a unor prelucrări, iar în funcție de modul în care este controlat procesul repetitiv există mai multe variante:
  - *Prelucrare repetitivă cu contor.* Se utilizează când se cunoaște de la început de câte ori trebuie executat corpul ciclului, structura generală a unei astfel de prelucrări fiind:

    `pentru` ⟨ contor ⟩ = ⟨valoare inițială ⟩, ⟨valoare finală⟩ `execută` ⟨ prelucrare ⟩

    In limbajele de programare C și Pascal, această prelucrare corespunde instrucțiunii `for` (varianta din limbajul C este mai generală, permițând specificarea unor condiții de continuare/oprire a prelucrării repetitive care nu se bazează exclusiv pe valoarea variabilei contor).

– *Prelucrare repetitivă condiționată anterior.* Se utilizează pentru a specifica prelucrări repetitive care se efectuează cât timp o anumită condiție (de continuare), specificată printr-o expresie relațională sau o expresie logică, este adevărată:

`cât timp` ⟨ condiție ⟩ `execută` ⟨ prelucrare ⟩

In cazul în care condiția de continuare este falsă de la început, prelucrarea specificată în corpul ciclului nu se execută niciodată. In limbajele de programare C și Pascal, această prelucrare corespunde instrucțiunii `while`.

– *Prelucrare repetitivă condiționată posterior.* Se utilizează pentru a specifica prelucrări repetitive care se efectuează până când o anumită condiție (de oprire) devine adevărată:

`repetă` ⟨ prelucrare ⟩ `până când` ⟨ condiție⟩

Prelucrarea corespunde instrucțiunii `repeat ... until` din limbajul Pascal și este corelată cu instrucțiunea `do ... while` din limbajul C (cu observația că include o condiție de continuare). Este utilă atunci când prelucrarea din corpul ciclului trebuie executată cel puțin o dată.

Probleme

1. Fie n un număr natural nenul. Descrieţi algoritmi pentru:

   (a) Determinarea sumei tuturor cifrelor lui n. De exemplu, pentru $n = 26326$ se obţine valoarea 19.

   (b) Determinarea valorii obţinute prin inversarea cifrelor numărului n. De exemplu, pentru valoarea 26326 se obţine valoarea 62362.

   (c) Determinarea tuturor cifrelor binare ale lui n.

   (d) Determinarea tuturor divizorilor proprii ai lui n.

   (e) Afișarea mesajului „da” dacă cea mai semnificativă cifră a unui număr n este strict mai mare decât cifra unităţilor sau mesajul „nu” în caz contrar. De exemplu, pentru $n = 5832$ se va afişa „da” pentru că $5 > 2$, iar pentru $n = 4539$ se va afişa „nu”.

2. *Cifra destinului.* Cifra destinului este cifra obţinută prin adunarea cifrelor ce intervin în data naşterii; adunarea cifrelor rezultatului obţinut se repetă până se ajunge la o singură cifră. De exemplu, pentru 01.02.1999 se obţine: $1+2+1+9+9+9 = 31 \rightarrow 3+1 = 4$. Descrieţi un algoritm care calculează cifra destinului pornind de la data nașterii specificată prin cele trei valori (zi, lună, an).

3. *Numere asemenea.* Două numere naturale sunt asemenea dacă scrierile celor două numere în baza 10 au aceleași cifre. De exemplu, numerele 23326 și 623 sunt asemenea, deoarece mulțimea cifrelor este aceeași ($\{2, 3, 6\}$). Descrieţi un algoritm care preia o pereche de numere naturale și determină dacă sunt sau nu asemenea.

4. *Conversie între baze de numeraţie.* Se consideră că numărul natural nenul n este reprezentat în baza de numeraţie b_1 ($2 \leq b_1 \leq 10$) şi se doreşte construirea numărului natural m care reprezintă aceeaşi valoare, însă în baza b_2 ($2 \leq b_2 \leq$

10). Atât numărul iniţial cât şi cel convertit sunt specificate prin variabile întregi (nu este posibilă utilizarea unui tablou pentru stocarea cifrelor). De exemplu, pentru $n = 2210$ şi $b_1 = 3, b_2 = 5$ se obţine $m = 300$.

5. *Reguli de codificare.* Se consideră un număr natural n constituit din $k \geq 2$ cifre ($n = c_k c_{k-1} \ldots c_1$) şi se pune problema codificării lui n prin schimbarea poziţiei unor cifre fără a stoca cifrele lui n într-un tablou. Descrieţi algoritmi pentru următoarele două variante de codificare:

   (a) *Permutare circulară a cifrelor.* Fiind dată o valoare p ($1 \leq p \leq k-1$), să se construiască numărul $m = c_p c_{p-1} \ldots c_1 c_k c_{k-1} \ldots c_{p+1}$. De exemplu, pentru $n = 45612$ şi $p = 2$ se obţine $m = 12456$.

   (b) *Interschimbare a două cifre.* Fiind date două valori p_1 şi p_2 ($1 \leq p_2 \leq p_1 \leq k-1$) să se construiască numărul obţinut prin interschimbarea cifrelor de pe poziţiile indicate de p_1 şi p_2 (din $n = c_k c_{k-1} \ldots c_{p_1} \ldots c_{p_2} \ldots c_1$ se obţine $m = c_k c_{k-1} \ldots c_{p_2} \ldots c_{p_1} \ldots c_1$), toate celelalte cifre rămânând pe poziţia lor iniţială. De exemplu, pentru $n = 45612$, $p_1 = 4$, $p_2 = 2$ se obţine $m = 41652$.

6. Pentru un număr natural n, să se scrie un algoritm care determină numărul maxim de divizori pe care îi are un număr din mulţimea $\{2, 3, \ldots, n\}$. De exemplu, dacă $n = 12$, numărul maxim de divizori este 5 (numărul 12 are 5 divizori: $2, 3, 4, 6, 12$).

7. Descrieţi un algoritm care, pentru două numere naturale nenule a şi b, determină numitorul şi numărătorul fracţiei ireductibile egale cu $\frac{a}{b}$.

8. Să se descrie un algoritm care verifică dacă un număr n este prim sau nu, folosind teorema lui Wilson. (Condiția necesară și suficientă ca un număr natural $p, p > 1$, să fie prim este ca $(p-1)! + 1$ să fie divizibil cu p.)

9. Să se descrie un algoritm care afişează toate numerele prime dintr-un interval dat $[a, b]$, a, b numere naturale, $a < b$.

10. Se consideră un număr natural $n > 2$. Descrieţi un algoritm care să afișeze toate numerele x mai mici decât n, care au proprietatea că $x - 1$ și $x + 1$ sunt numere prime. Exemplu: pentru $n = 43$, se vor afișa numerele: 4 6 12 18 30 42.

11. Scrieți un algoritm care afişează descompunerea unui număr natural n în factori primi. De exemplu, pentru $n = 18$, se va afişa $2\hat{}1 \times 3\hat{}2$.

12. Să se scrie un algoritm care, pentru numărul natural n dat, determină sumele:

   (a) $S = 1 \times 2 + 2 \times 3 + 3 \times 4 ... + n \times (n+1)$

   (b) $S = 1 \times n + 2 \times (n-1) + 3 \times (n-2) + ... + n \times 1$

13. Estimaţi cu precizia $\varepsilon > 0$ limitele șirurilor următoare (pentru punctele (a) şi (b) se consideră că x este o valoare dată din intervalul $(0, 1)$):

   (a) $s_n = \sum_{i=0}^{n} \frac{(-1)^i x^{2i+1}}{(2i+1)!}$

(b) $s_n = \sum_{i=0}^{n} \frac{(-1)^i x^{2i}}{(2i)!}$

(c) $s_n = \sum_{i=1}^{n} \frac{1}{(2i+1)^2}$

Indicaţie. Calculul se opreşte atunci când diferenţa dintre doi termeni consecutivi ai șirului este mai mică decât constanta $\varepsilon(|s_n - s_{n-1}| < \varepsilon, \varepsilon = 0.001)$.

14. Să se scrie un algoritm care determină dacă un număr natural dat m face parte din șirul lui Fibonacci sau nu. Șirul lui Fibonacci este definit prin relațiile: $f(0) = 1, f(1) = 1, f(n) = f(n-1) + f(n-2)$, pentru $n \geq 2$. Nu se vor folosi tablouri.

1.2 Descompunerea unei probleme în subprobleme

Tematică

Algoritmi de prelucrare a datelor simple şi exemple de utilizare a subprogramelor (funcţii şi proceduri) și a diferitelor tipuri de parametri (parametri de intrare, ieșire, intrare/ieșire) și variabile (locale, globale). Tipuri de prelucrări:

- Prelucrări asupra cifrelor (extragerea şi analiza cifrelor unui număr natural)
- Prelucrări ale unor secvenţe de valori preluate secvenţial (fără să fie stocate în structuri de tip tablou)
- Prelucrări asupra unor mulțimi de puncte în plan, date prin coordonatele lor carteziene
- Evaluarea unui polinom specificat prin valorile coeficienților
- Prelucrări cu numere complexe

Probleme

1. Se asociază unui număr natural n valoarea $A(n)$ ca fiind suma factorialelor cifrelor lui n (de exemplu, pentru $n = 318,\ A(n) = 3! + 1! + 8!$).

   (a) Pentru o valoare n dată să se calculeze $A(n)$. Calculaţi numărul de operaţii de înmulţire efectuate. Propuneţi o variantă care să utilizeze un număr cât mai mic de operaţii de înmulţire.

   (b) Pentru o valoare naturală k ($k < 9999$) să se determine cel mai mic număr natural n cu proprietatea că $A(n) = k$.

2. Determinaţi cel mai mic şi cel mai mare număr constituit din k cifre ($k < 5$) având proprietatea că suma cifrelor este S. Date de test: $k = 3, S = 25; k = 4, S = 33; k = 4, S = 12$.

3. Se pune problema generării unei valori naturale n pornind de la 0 şi folosind doar următoarele două operaţii: (i) dublarea valorii curente; (ii) incrementarea valorii curente. De exemplu, $5 = (0 + 1) \times 2 \times 2 + 1$ se poate obţine aplicând secvenţa de operaţii: incrementare; dublare; dublare; incrementare, iar $12 = (((0+1) \times 2+1) \times 2) \times 2$ se obţine aplicând: incrementare, dublare, incrementare, dublare, dublare.

   (a) Pentru o valoare naturală n determinaţi o secvenţă de operaţii care permite generarea valorii.

   (b) Determinaţi numărul minim de operaţii de incrementare şi dublare care permit generarea valorii n.

4. Se citeşte o secvenţă de n perechi de valori reale $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ care reprezintă coordonatele vârfurilor unui poligon convex $P_1P_2 \ldots P_n$ (punctul P_i are coordonatele (x_i, y_i)) specificate în ordine trigonometrică. Să se determine:

   (a) Perimetrul poligonului.

   (b) Aria poligonului.

   (c) Coordonatele vârfurilor celui mai mic dreptunghi, având laturile paralele cu axele de coordonate, care conţine poligonul.

   (d) Numărul efectiv de laturi ale poligonului (se presupune că pot exista mai mult de două puncte consecutive coliniare).

   Date de test: $P_1(2, 5)P_2(1, 4)P_3(1, 2)P_4(2, 1)P_5(5, 1)P_6(8, 1)P_7(9, 3)P_8(8, 5)P_9(5, 5)$

5. Se consideră polinomul de grad n cu coeficienţi reali: $P(z) = c_n z^n + c_{n-1} z^{n-1} + \cdots + c_1 z + c_0$ şi se pune problema calculului valorii polinomului pentru un număr complex $z = a + bi$, specificat prin partea sa reală a şi coeficientul părţii sale imaginare b. Valorile coeficienţilor se citesc succesiv începând cu c_n.

   Date de test: $z = 2 + 3i, P(z) = 2z^3 - 3z^2 + z - 6$.

6. Se citeşte succesiv o secvenţă de n valori reale care conţine cel puţin o valoare pozitivă. Să se determine indicele de început şi indicele de sfârşit ale unei subsecvenţe de elemente consecutive care satisface:

   (a) Este cea mai lungă subsecvenţă crescătoare din secvenţa dată.

   (b) Suma elementelor subsecvenţei este maximă (în raport cu sumele altor subsecvenţe din secvenţă).

   Date de test: $-1, 2, 1, -1, 3, 4, 5, -3, 1, 4$
   $2, -3, 1, 3, 4, -2, 1, 2, 3, -5$

7. Se consideră un serviciu web la care utilizatorii se conectează/deconectează şi se pune problema determinării numărului maxim de utilizatori conectaţi simultan pornind de la o secvenţă de semnale de forma: 1 (s-a conectat un utilizator), 0 (s-a deconectat un utilizator). De exemplu, pentru secvenţa $1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0$ numărul maxim de utilizatori conectaţi este 5.

8. O secvenţă de n valori reale preluate succesiv (fără a fi stocate într-un tablou): $x_0, x_1, \ldots, x_{n-1}$ trebuie "netezită" prin mediere folosind o "fereastră" de dimensiune 3, adică, pornind de la valorile din secvenţă, se afişează valorile $y_1, y_2, \ldots, y_{n-2}$ care au proprietatea că $y_i = (x_{i-1} + x_i + x_{i+1})/3$.

9. Se consideră relaţia de recurenţă care defineşte o secvenţă de tip Collatz:

$$C(n) = \begin{cases} \text{n/2} & \text{dacă } n \text{ este par} \\ \text{3n+1} & \text{dacă } n \text{ este impar} \end{cases}$$

   Conjectura Collatz afirmă faptul că pentru orice valoare de pornire n, secvenţa va ajunge la 1. Pentru un număr natural n se defineşte $L(n)$ ca fiind lungimea secvenţei, adică numărul de elemente din secvenţă până la întâlnirea valorii 1 (inclusiv valoarea 1). Două numere, n_1 şi n_2, se consideră Collatz-echivalente dacă $L(n_1) = L(n_2)$. Pentru două valori naturale, a şi b ($a < b$), să se determine toate perechile de numere Collatz-echivalente din mulțimea $\{a, a+1, \ldots, b\}$.

10. Se consideră o funcţie integrabilă $f : [a, b] \to R$ şi se pune problema aproximării integralei funcţiei f pe intervalul $[a, b]$ folosind o sumă Riemann "la stânga" cu pasul $h(n) = (b-a)/n : S(n) = h(n)(f(a)+f(a+h(n))+\cdots+f(b-h(n)))$. Fiind date limitele intervalului, să se calculeze $S(n^*)$ (n^* este prima valoare pentru care $|S(n^*) - S(n^* - 1)| < 0.001$).

1.3 Prelucrarea tablourilor unidimensionale

Tematică

- Declararea și parcurgerea tablourilor unidimensionale
- Ștergeri și inserări de elemente
- Verificarea unor proprietăți
- Sortarea tablourilor
- Interclasarea tablourilor
- Secvențe de vectori

Sumar teoretic

Un tablou unidimensional este o structură de date care ne permite să stocăm o serie de valori de același tip. În general, primul element din tablou se găsește pe poziția 0, dar există limbaje de programare care încep numerotarea de la 1. În continuare, vom considera poziția 0 ca primă poziție în tablou. În acest caz, ultimul element se va găsi pe poziția $N-1$, unde N este numărul total de elemente din tablou. Mai departe vom utiliza limbajul *C++*.

Pentru a defini o variabilă de tip tablou în *C/C++* se folosește următoarea sintaxă:

```
tip tablou[N];
```

unde *tip* reprezintă un tip de date, *tablou* este numele variabilei, iar N este *dimensiunea* tabloului (numărul de elemente din tablou).

Exemple:

```
char sir[10]; // un sir de 10 caractere
int note_elev[5]; // un tablou de 5 de numere intregi
float medii_elevi[28] // un tablou de 28 de numere reale
```

Pentru a seta sau accesa valorile din tablou, ne folosim de parantezele drepte ([ ]), între care folosim poziția din tablou. În continuare vom folosi termenul **index** și poziție interschimbabil.

În exemplul de mai jos vom seta pe pozițiile 0, 1 și 2 valorile 7, 8 și 9, respectiv. Mai departe vom afișa nota de pe poziția 1.

```
int note[3];
note[0] = 7;
note[1] = 8;
note[2] = 9;
cout << note[1];
```

Pentru a afișa toate notele, putem folosi o instrucțiune repetitivă pentru a parcurge întregul tablou:

```
int i;
for(i = 0; i < 3; i++){
   cout << note[i];
}
```

Inițializarea tablourilor

În funcție de locul unde tabloul este definit în program, acesta poate fi inițializat implicit cu diferite valori. Dacă tabloul este o variabilă globală, atunci acesta va fi inițializat cu valoare 0 pe fiecare poziție din tablou. Dacă tabloul este o variabilă locală, chiar și a funcției *main()*, atunci nu avem nici o garanție cu privire la valorile cu care este inițializat. Aceste valori vor fi *valori reziduale* care sunt prezente în memorie atunci când tabloul este inițializat. Pentru a inițializa toate elementele unui tablou care este variabilă locală, este necesară parcurgerea și setarea valorii fiecărui index din tablou.

Exemplu:

```
int global[5]; // toate elemente sunt initializate cu 0
int main(){
   int local[5];
   int i;
   for(i = 0; i < 5; i++){
```

```
      cout << global[i]; // in mod cert valoarea va fi 0
      cout << local[i] // imposibil de cunoscut valoarea afisata
      local[i] = 0; // este necesara initializarea
   }
}
```

Alternativ, se poate folosi *inițializarea inline* cu ajutorul acoladelor ({}):

```
int main(){
    int local[5] = {1,2,3,4,5}; // initializarea explicita a fiecarui
       element
    int zerouri[5] = {0}; // initializarea tuturor elementelor cu 0
   int primele[10] = {1,2,3}; // primele 3 pozitii au valorile 1, 2 si 3,
      iar restul elementelor vor avea valoarea 0.
}
```

Transmiterea tablourilor ca argumente ale funcțiilor

Ne reamintim că în limbajul *C++* putem transmite argumentele funcțiilor în două moduri: prin *valoare* sau prin *referință* (alăturând simbolul & argumentului). Tablourile sunt, în schimb, transmise folosind pointeri, prin urmare nu este nevoie de simbolul &. În practică, este transmisă adresa primului element din tablou, celelalte aflându-se în continuarea acestuia. Din acest motiv, următoarele definiții de funcții sunt echivalente:

```
(i) void func(int arr[10]);
(ii) void func(int arr[]);
 //dimensiunea nu este obligatorie
(iii) void func(int *arr);
 //pointer in loc de tablou, ambele determina transmiterea adresei primului
    element.
```

Prin urmare, un tablou trimis ca argument către o funcție va putea fi modificat de către aceasta. Următorul exemplu folosește o funcție pentru a citi de la tastatură elementele unui tablou.

```
void citire_note(int note[], int N){
   int i;
   for(i = 0; i < N; i++){
      cout << "Introduceti nota " << i << ": ";
      cin >> note[i];
   }
}

int main(){
    int note[5];
    citire_note(note, 5);
}
```

Probleme

1. *Parcurgere şi construire tablouri.* Se consideră două tablouri, A și B, având fiecare n elemente numere întregi. Să se scrie câte o funcţie/procedură corespunzătoare următoarelor probleme:

   (a) Considerăm că A şi B au aceeaşi lungime: se cere să se construiască un tablou C care conţine elementele comune, care apar atât în A cât şi în B pe aceeaşi poziţie, dacă nu există acelaşi element se va pune 0 în tabloul C.

   (b) Calculează suma elementelor din A care apar şi în B pe aceeaşi poziţie.

   (c) Construieşte un tablou D care conţine pe poziţia i suma elementelor corespondente din A şi B. Dacă au dimensiuni diferite, D va conţine o copie a elementului din tabloul mai lung.

2. *Conversii între baze de numeraţie (utilizând tablouri).*

   (a) Se consideră că numărul natural nenul n este reprezentat în baza 10 şi se doreşte construirea tabloului de caractere care sa conţină cifrele numărului.

   (b) (generalizare a) Se consideră că numărul natural nenul n este reprezentat în baza 10 şi se doreşte construirea numărului m care reprezintă aceeaşi valoare, însă în baza b ($2 \leq b \leq 9$). Numărul m va fi reprezentat printr-un șir de caractere.

   (c) Se consideră un tablou (B) de întregi care conţine doar 0 sau 1. Tabloul conține reprezentarea în baza 2 a unui număr natural şi se pune problema construirii unui tablou (H) care conţine reprezentarea în baza 16. De exemplu, pentru $B = 10110011$ se obţine $H = B3$.

   (d) Se consideră un şir de caractere care conţine reprezentarea în baza 16 a unui număr (alfabetul folosit este '0', '1',..., 'A',...,'F') şi se pune problema determinării valorii corespunzătoare în baza 10. De exemplu, pentru $B9F$ se obţine 2975.

3. *Reprezentarea în complement față de* 2. Numerele întregi negative sunt reprezentate intern prin regula complementului față de 2 care constă în aplicarea următoarelor etape: *i*) se determină reprezentarea în binar a numărului fără semn și se complementează toți biții (un bit 0 se transformă în 1, iar un bit 1 se transformă în 0); *ii*) se adună 1 la numărul anterior (folosind regulile de adunare în baza 2). Ceea ce se obține este reprezentarea internă a numărului întreg negativ. Dacă se dă o reprezentare internă hexazecimală ($0xA2B3C4D5$), prin examinarea bitului de semn (cel mai din stânga) se decide dacă numărul este negativ sau nu. Dacă numărul este negativ, atunci se vor aplica aceleași reguli pentru a determina valoarea numărului. Să se scrie o funcție care să compare două numere întregi pe 32 de biți pentru care se cunoaște reprezentarea internă în hexazecimal. Funcția va returna

   (a) -1 dacă primul număr este mai mic decât al doilea;

   (b) 0 dacă sunt egale;

   (c) 1 dacă primul număr este mai mare decât al doilea.

4. Definiți un tablou cu numele de numere reale (tip double) de dimensiune 100. Definiți un ciclu ce va popula elementele tabloului cu următoarea secvență:

   $$1/(2*3*4), 1/(4*5*6), 1/(6*7*8), \ldots \text{ până la } 1/(200*201*202)$$

   Presupunând că numele tabloului este `data`, parcurgeți tabloul pentru a calcula următoarea expresie:

   $$\texttt{data}[0] - \texttt{data}[1] + \texttt{data}[2] - \texttt{data}[3] + \cdots - \texttt{data}[99]$$

   Înmulțiți rezultatul obținut anterior cu 4.0 și adăugați 3.0 și afișați rezultatul. Recunoașteți valoarea obținută?

5. *Gigel, Ionel și Călin,* trei studenți de la Informatică au pornit un StartUp de succes împreună. Fiecare știe câteva limbaje sau tehnici de programare: *Gigel știe n limbaje, Ionel știe m, iar Călin k.* Înainte de a porni startup-ul cu propriul lor proiect, pentru a face bani, s-au hotărât să participe la orice proiect pe care l-ar putea face de pe `freelancer.com`, în orice combinație posibilă. Fiecare astfel de proiect necesită o mulțime de cunoștințe prezentate foarte clar în descrierea sa. Să presupunem că se cunosc cerințele pentru 30 de proiecte, să se scrie un program care să stabilească alocarea studenților la proiecte în așa fel încât să nu se suprapună competențele (pentru fiecare proiect se stabilește dacă vor intra singuri, câte doi sau toți trei împreună).

   Exemplu: Gigel știe {C, C#, OOP și Java}, Ionel {C, Python, C++, OOP} și Călin {C++, OOP, Ruby}. Dacă se cunosc cerințele pentru

   `Proiect 1:` {OOP, C, Ruby, Python} → este nevoie de toți 3 pentru acest proiect.

   `Proiect 2:` {C#, Java} → Gigel îl poate face singur.

6. *Planificare service auto.* Într-un service auto există 5 elevatoare și 5 mecanici. Într-o lună se repară în acel service, în medie, 200 de mașini. Șeful de service, Gigel, este cel care planifică fiecare mașină știind pentru fiecare intervenție când intră în service și cât durează reparația. Se cunosc datele din planificare, când intră fiecare mașină și cât durează intervenția (vor fi generate aleator), și mai știm că Gigel, atunci când planifică, se uită pe rând care elevator este liber și îl marchează în planificare. Dacă nu are niciunul disponibil, atunci, împreună cu clientul, alege o altă dată și oră disponibilă. Întrebarea la care vrem să răspundem este care din cele 5 elevatoare va fi folosit cel mai mult în luna ce urmează și care vor fi mașinile reparate, în ordinea cronologică a reparațiilor.

   Exemplu: Presupunem că în loc de 20 zile a câte 8 ore pe zi ne gândim la 160 de ore pe lună. Ca intrare vom primi 2 seturi de numere care specifică ora la care începe și cât durează întervenţia pentru o maşină $(1, 1, 5, 10, 7)$ şi $(4, 6, 4, 3, 4)$:

Maşina	Ora start	Durata
1	1	4
2	1	6
3	5	4
4	10	3
5	7	4

vom planifica astfel:

$1,\ 1+4 \to 5, 4+5 \to 10,\ 10+3$

$1,\ 1+6 \to 7,\ 7+4$

7. *Spălătorie auto self-service.* La colţul străzii există o spălătorie auto self-service cu 5 locuri. Istoricul unei zile arată în felul următor:

   - timpii de intrare în spălătorie : $[8:00, 8:05, 8:05, 8:08, 8:32, 8:37, 8:43, 8:45 \dots]$
   - numărul de fise folosite: $[2, 3, 4, 2, 1, 2, 3, 4 \dots]$

   Se cere să se genereze întâi datele de intrare (istoricul unei zile la spălătorie):

   (a) Se va genera ora și minutul pentru fiecare dintre timpii de intrare apoi se vor ordona crescător după oră și minute.

   (b) Se va genera aleator numărul de fise folosite, un număr între 1 și 5.

   Știind că o fisă durează 2 minute la care se va adăuga 1 minut, timpul pierdut pentru pregătire, se cere să se implementeze funcții/proceduri corespunzătoare următoarelor probleme:

   (c) Ştiind că o fisă costă 2.5 lei să se calculeze profitul zilei.

   (d) Să se determine, pentru fiecare maşină, ora la care iese din spălătorie.

   (e) Să se determine pentru fiecare maşină dacă trebuie să aştepte sau nu ca să intre într-o boxă.

   (f) Să se determine care este cel mai lung timp de aşteptare.

8. *Clasament WTA.* În acest moment conform `https://www.tenisdecamp.ro/clasament-wta-simplu/` în clasamentul WTA la tenis feminin primele 10 jucătoare au următoarele puncte:

1	Australia	Ashleigh Barty	8017
2	Czech Republic	Karolina Pliskova	5940
3	Romania	Simona Halep	5561
4	Japan	Naomi Osaka	5496
5	Ukraine	Elina Svitolina	5075
6	Canada	Bianca Vanessa Andreescu	4935
7	Switzerland	Belinda Bencic	4675
8	Czech Republic	Petra Kvitova	4436
9	USA	Serena Williams	4215
10	Holland	Kiki Bertens	4165

Presupunem că în fiecare lună va fi câte un concurs de GrandSlam și ştim că pentru GrandSlam se pot câștiga următoarele puncte

	W	F	SF	QF	R16	R32	R64	R128	Q	Q3	Q2	Q1
Points	2000	1300	780	430	240	130	70	10	40	30	20	2

(W=winner, F=Finalist, ...)

Fiecare dintre jucătoarele din top 10 ar putea să câștige oricare meci și oricare turneu.

S-a terminat turneul, se cunosc rezultatele şi ce s-a întâmplat anul trecut:

		Anul trecut	Turneul actual
1	Ashleigh Barty	W	F
2	Karolina Pliskova	QF	W
3	Simona Halep	F	SF
4	Naomi Osaka	SF	SF
5	Elina Svitolina	QF	QF
6	Bianca Vanessa Andreescu	R16	SF
7	Belinda Bencic	QF	QF
8	Petra Kvitova	SF	QF
9	Serena Williams	R32	QF
10	Kiki Bertens	SF	QF

Având în vedere că fiecare punct câştigat anterior trebuie apărat, se pune întrebarea dacă Simona Halep va fi sau nu numărul 1 după acest turneu?

9. *Analiză zaruri.* Bogdan are 2 zaruri cu câte 6 fețe fiecare, unul albastru și unul roșu. Le aruncă pe o masă de 100 de ori și își notează valorile obținute pentru fiecare zar în parte (în câte un tablou cu 100 de elemente pentru fiecare zar). Să se decidă care dintre cele 2 zaruri este cel mai echilibrat zar? Cel albastru sau cel roşu?

   Notă: Se va folosi formula entropiei lui Shannon pentru a determina cantitatea de informație existentă în fiecare zar. Zarul cu entropia mai mare este considerat mai echilibrat. Dacă $(f_1, f_2, \ldots, f_6)$ reprezintă frecvenţele relative asociate valorilor obţinute la aruncarea unui zar, regula de calcul a entropiei este $-f_1 \times \log_2(f_1) - f_2 \times \log_2(f_2) - \cdots - f_6 \times \log_2(f_6)$, entropie mai mare înseamnă mai echilibrat.

10. *Sufix şi prefix comun.* Fiind date două tablouri $(a_1, a_2, \ldots, a_m)$ şi $(b_1, b_2, \ldots, b_n)$, cu numere întregi, să se stabilească numărul maxim de elemente comune aflate la sfârşitul lui a şi începutul lui b (de exemplu, $a = (1, 4, 3, 5, 6, 7, 2)$ şi $b = (6, 7, 2, 5, 1, 8)$ au 3 elemente comune, pe $6, 7$ şi 2).

11. *Date de naştere.* Se consideră o listă care conţine datele de naştere ale locuitorilor unui oraş (lista poate fi reprezentată printr-un tablou conţinând structuri sau prin 3 tablouri corespunzătoare celor 3 componente ale date: (zi, lună, an)). Să se determine:

    (a) cea mai "mică" dată de naştere (corespunzătoare celui mai vârstnic locuitor);

    (b) cea mai "mare" dată de naştere (corespunzătoare celui mai tânăr locuitor);

    (c) cea/cele mai frecventă/frecvente dată/date de naştere (cea/cele care apare/apar de cele mai multe ori în listă);

    (d) dacă există vreo zi în perioada (1 ianuarie 1918 – 31 decembrie 2017) în care nu e născut niciun locuitor.

1.4 Prelucrarea tablourilor bidimensionale

Tematică

- Construire, parcurgere şi transformare
- Operaţii cu matrici (adunare, înmulţire)
- Verificare proprietăţi (simetrie, matrici triunghiulare, matrici diagonale etc.)
- Aplicații

Sumar teoretic

În descrierea elementelor teoretice am folosit Capitolul 5 din manualul de informatică scris de Sorin Tudor[1].

Interpretare matematică

Fie $A_{n_i} = \{1, 2, ..., n_i\}$ mulțimea primelor n_i numere naturale. Fie $M = A_{n_1} \times A_{n_2} \times ... \times A_{n_k}$ produsul cartezian a k astfel de mulțimi.

Se numește *tablou* o funcție $f : M \to T$, unde T este o mulțime oarecare. k este dimensiunea tabloului. Dacă $k = 1$ tabloul se mai numește și *vector* și are n_1 componente. Dacă $k = 2$ tabbloul se numește *matrice* și are $n_1 \times n_2$ elemente.

Exemplul 1.4.1. Fie vectorul $v = (1, 2, 5, -6, 7)$. Atunci $k = 1$, $n_1 = 5$, $T = \mathbb{N}$ iar elementele vectorului sunt $v_1 = 1$, $v_2 = 2$, $v_3 = 5$, $v_4 = -6$, $v_5 = 7$.

Exemplul 1.4.2. Fie matricea A cu m linii și n coloane cu elemente din $\mathbb{R}$. A este numele tabloului, elementele sale sunt $a_{11}, a_{12}, a_{13}, ...a_{1n}, ..., a_{m1}, a_{m2}, ..., a_{mn}$, iar a_{ij} este elemetul de pe linia i, coloana j.

Tablouri în C/C++

În C/C++ tipul tablou se declară astfel: `tip nume`$[n_1]$`[`n_2`]...[`n_n`]`.

Exemplul 1.4.3. Fie `int v[100]`. Am declarat un vector cu 100 de elemente numere întregi.

Exemplul 1.4.4. Fie `float a[10], b[50]`. Am declarat doi vectori ambii cu elemente de tip `float`, primul cu 10 elemente, al doilea cu 50 de elemente.

Exemplul 1.4.5. Fie `int a[10][5]`. Am declarat o matrice cu elemente de tip `float`, cu 10 linii și 5 coloane.

Limbajul C/C++ nu ne permite să declarăm o variabilă tablou cu număr varuabil de componente, de aceea la declarație se declară un tablou cu un număr maxim de elemente.

[1]Sorin Tudor, *Informatica,*, Manual pentru clasa a IX-a Intensiv sau clasa a X-a Real (Var. C++), L&S Info-Mat, 2012.

Remarca 1.4.1. În C/C++ idexarea elementelor în tablou începe cu poziția 0.

Operațiile de bază cu matrici sunt citirea, respectiv scrierea.

```
void citesteMatrice(int a[10][10], int lin, int col)
{
    int i, j;
    for(i=0; i < lin; i++)
    {
        for(j=0; j < col; j++)
        {
            cout << "Dati elementul [" << i <<"][" << j << "] ";
            cin  >> a[i][j];
        }
    }
}

void afiseazaMatrice(int m[10][10], int lin, int col)
{
    int i,j;
    for(i = 0; i < lin; i++)
        {
            for(j = 0; j < col; j++)
                cout << m[i][j] << " ";
            cout << "\n";
        }
}
```

Remarca 1.4.2. Știm că transmiterea parametrilor funcțiilor se poate face prin sau referință.Tablourile fac excepție de la această regulă, o funcție putând să modifice conținutul unui șir care i-a fost transmis ca parametru. Motivul este că funcția primește de fapt adresa primului element din tablou, și accesează astfel toate elementele originale. Astfel că *nu este nevoie de operatorul & pentru tablouri.*

```
main()
{
    int m1[10][10], m2[10][10, lin1, col1, lin2, col2;
    cout << "Nr. de linii al primei matrici: "; cin >> lin1;
    cout << "Nr. de coloane al primei matrici: "; cin >> col1;
    cout << "Nr. de linii al celei de a doua matrici: "; cin >> lin2;
    cout << "Nr. de coloane al celei de a doua matrici: "; cin >> col2;
    cout << "Introduceti elementele primei matrice\n";
    citesteMatrice(m1, lin1, col1);
    cout << "Introduceti elementele celei de a doua matrice\n";
    citesteMatrice(m2, lin2, col2);
}
```

Probleme

1. Să se verifice dacă o valoare x se găseşte într-o matrice cu m linii și n coloane.

   Indicație: Se parcurge matricea pe linii, apoi pe coloane, iar la prima întâlnire a elementului căutat ne oprim.

2. Calculați suma și produsul a două matrici.

   Indicație: În cazul adunării, cele două matrici trebuie să aibă aceeași dimensiune. În cazul produsului, numărul de coloane ale primei matrici trebuie să fie egal cu numărul de linii ale celei de a doua. Matricea produs va avea numărul de linii al primei matrici și numărul de coloane al celei de a doua.

3. Calculați suma elementelor de pe diagonala principală și produsul elementelor de pe diagonala secundară.

   Indicație: Pentru rezolvarea problemei, trebuie să ne asigurăm că matricea este pătratică. Elementele de pe diagonala principală au indicii (i, i), iar cei de pe diagonala secundară $(i, n - i + 1)$, unde n este dimensiunea matricii.

4. Să se determine transpusa unei matrici.

   Indicație: Transpusa unei matrici se obține interschimbând liniile cu coloanele.

5. Să se elimine linia i şi coloana j dintr-o matrice cu m linii și n coloane.

6. Determinați dacă o matrice este simetrică (față de diagonala principală, respectiv secundară), triunghiulară (superior, respectiv inferior), diagonală.

7. Să se verifice dacă o matrice pătratică de dimensiune $n \times n$ este pătrat magic (suma elementelor de pe fiecare linie, coloană, diagonala principală şi diagonala secundară este aceeaşi şi este egală cu $\frac{n^2(n^2+1)}{2}$ şi memorează toate valorile de la 1 la n^2).

   Exemplu: $\begin{pmatrix} 6 & 1 & 8 \\ 7 & 5 & 3 \\ 2 & 9 & 4 \end{pmatrix}$.

   Contraexemple: $\begin{pmatrix} 6 & 0 & 8 \\ 7 & 5 & 3 \\ 2 & 9 & 4 \end{pmatrix}$, $\begin{pmatrix} 5 & 10 & 5 \\ 11 & 5 & 4 \\ 3 & 3 & 14 \end{pmatrix}$.

8. Se citesc de la tastatură dimensiunile unei matrici și elementele ei. Se cere transformarea matricii prin inversarea ordinii elementelor în cadrul fiecărei linii.

   Exemplu: Pentru matricea $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 2 & 1 \\ 2 & 3 & 4 & 1 & 8 \end{pmatrix}$ se obține $\begin{pmatrix} 5 & 4 & 3 & 2 & 1 \\ 1 & 2 & 1 & 4 & 3 \\ 8 & 1 & 4 & 3 & 2 \end{pmatrix}$.

9. Pentru o matrice cu n linii și m coloane, care memorează numere întregi, modificați ordinea elementelor astfel încât parcurgând matricea de sus în jos şi de la stânga la dreapta aceasta să conţină mai întâi elementele negative, apoi pe cele pozitive. Se va analiza atât varianta în care se poate utiliza un tablou adițional (cu $m \times n$ elemente), cât și varianta în care nu se utilizează tablou adițional.

Exemplu: Pentru matricea $\begin{pmatrix} -5 & 6 & -8 & 3 \\ 2 & -9 & 1 & -6 \\ 8 & 1 & -8 & -2 \end{pmatrix}$ se obține $\begin{pmatrix} -5 & -9 & -8 & -8 \\ -6 & -2 & 1 & 2 \\ 8 & 1 & 3 & 6 \end{pmatrix}$.

10. Se consideră o reţea de n calculatoare şi o matrice pătratică de dimensiune n care conţine elemente din $\{0, 1\}$ completate astfel: $a[i, j] = 1$ dacă calculatorul i este conectat direct cu calculatorul j şi a$[i, j] = 0$ în caz contrar. Scrieţi algoritmi care verifică dacă:

    (a) reţeaua este complet conectată ($a[i, j] = 1$ pentru fiecare pereche (i, j) de valori diferite)

    (b) reţeaua este de tip inel ($a[i, i + 1] = a[i + 1, i] = 1$ pentru fiecare i din $1, \ldots, n - 1$ şi a$[n, 1] = a[1, n] = 1$, iar celelalte elemente sunt nule)

    (c) reţeaua este de tip stea (există un i^* cu proprietatea că $a[i, i^*] = a[i^*, i] = 1$, pentru orice $i \neq i^*$, iar celelalte elemente sunt nule).

    Notă: Considerăm că un calculator nu este conectat cu el însuși, astfel că $a[i, i] = 0$, pentru fiecare i.

11. Se consideră o imagine color de dimensiune $n \times n$ pixeli. Știind că fiecărui pixel îi corespund trei valori din mulțimea $\{0, 1, \ldots, 255\}$ (câte una pentru fiecare dintre cele trei canale de culoare - roșu, verde și albastru) propuneți o structură de date pentru stocarea imaginii. Descrieți un algoritm care:

    (a) transformă imaginea color într-o imagine pe nivele de gri folosind pentru fiecare pixel regula:

    $$\text{gri} = \frac{max(\text{roșu,verde,albastru}) + min(\text{roșu,verde,albastru})}{2}$$

    (b) construiește histograma (tabelul cu frecvențele corespunzătoare valorilor pixelilor) asociată imaginii pe nivele de gri

    (c) determină valoarea medie folosind histograma construită la punctul (b)

    (d) transformă imaginea pe nivele de gri în imagine alb-negru (alb are codul 1, negru are codul 0) folosind valoarea determinată la punctul (c) ca valoare de prag (dacă valoarea pixelului din imaginea pe nivele de gri este mai mică decât valoarea medie atunci valoarea pixelului în imaginea alb-negru este 0, altfel este 1)

    (e) verifică dacă imaginea alb-negru construită la punctul anterior conține pixeli negri pe diagonala principală și pe cea secundară și pixeli albi în rest.

    Indicație: Putem asimila o imagine cu 3 matrici, Mr – matricea pixelilor roșii, Mv – matricea pixelilor verzi, Ma – matricea pixelilor albaștri. Atunci structura de date este tuplul format din cele 3 matrici. Pentru rezolvarea punctului (a) se folosesc 2 funcții auxiliare de calcul al maximului/minimului dintre două numere, iar ca operație principală este adunarea de matrici. La punctul (b) se ține cont de faptul că tabloul de frecvențe are lungimea de 256. Punctele (b) și (c) sunt aplicații la adunarea a două matrici, conceptul de adunare a elementelor din matrice. Punctele (d) și (e) testează proprietăți ale matricilor.

12. Se consideră o matrice pătratică cu n linii și n coloane. Descrieți câte un algoritm pentru:

    (a) afișarea elementelor din matrice linie după linie

    (b) afișarea elementelor din matrice coloană după coloană

    (c) afișarea elementelor din matrice diagonală după diagonală (pornind de la elementul de pe prima linie ultima coloană și parcurgând fiecare diagonală paralelă cu diagonala principală pornind de sus în jos).

    De exemplu, pentru o matrice cu 4 linii și 4 coloane și elemente $a[i, j]$, acestea vor fi afișate în următoarea ordine: $a[1, 4], a[1, 3], a[2, 4], a[1, 2], a[2, 3], a[3, 4], a[1, 1], a[2, 2], a[3, 3], a[4, 4], a[2, 1], a[3, 2], a[4, 3], a[3, 1], a[4, 2], a[4, 1]$.

    (d) afișarea elementelor în spirală pornind de la elementul de pe prima linie și prima coloană și parcurgând matricea în sensul acelor de ceasornic.

    Pentru exemplul de mai sus, ordinea de afișare va fi: $a[1, 1], a[1, 2], a[1, 3], a[1, 4], a[2, 4], a[3, 4], a[4, 4], a[4, 3], a[4, 2], a[4, 1], a[3, 1], a[2, 1], a[2, 2], a[2, 3], a[3, 3], a[3, 2]$.

1.5 Algoritmi recursivi

Tematică

- Descrierea algoritmilor recursivi
- Tehnici recursive
- Proceduri și funcții recursive
- Analiza complexității algoritmilor recursivi
- Exemple

Probleme

1. Să se determine cel mai mare divizor comun a două numere naturale (cmmdc) folosind o implementare recursivă a algoritmului lui Euclid bazat pe scăderi repetate.

2. Să se determine cel mai mare divizor comun a două numere naturale (cmmdc) folosind o implementare recursivă a algoritmului lui Euclid bazat pe împărțiri repetate.

3. În matematică, factorialul unui număr natural n (folosit adesea în algebră, combinatorică și analiza matematică), notat cu $n!$, este produsul tuturor numerelor naturale mai mici sau egale cu n. Prin convenție, $0!$ este egal cu 1. Să se scrie o funcție recursivă pentru calculul valorii factorialului unui număr întreg dat.

4. Cu ajutorul unei funcții recursive, să se verifice dacă un număr natural este factorialul unui alt număr natural.

5. Șirul lui Fibonacci este un șir de numere naturale în care, cu excepția primilor doi termeni egali cu 1, ceilalți termeni reprezintă suma celor doi anteriori. Acest șir este unul dintre cele mai populare șiruri specificate printr-o relație de recurență.

   (a) Să se scrie o funcție recursivă care determină valoarea termenului din șir de un rang specificat.

   (b) Să se scrie o funcție recursivă care determină rangul primului termen care atinge sau depășește o valoare dată.

6. Se consideră un număr natural n. Scrieți câte o funcție recursivă pentru:

   (a) Calculul sumei cifrelor numărului n

   (b) Calculul produsului cifrelor nenule ale numărului n

7. Înțelegem prin răsturnatul unui număr, numărul obținut atunci când cifrele se parcurg în ordinea inversă celei naturale (cea mai din stânga cifră este cifra unităților, următoarea spre dreapta este cifra zecilor ș.a.m.d.). Să se scrie o funcție recursivă care generează răsturnatul unui număr natural primit ca parametru.

8. Scrieți o funcție recursivă care verifică dacă numărul natural primit ca parametru poate fi scris ca sumă de termeni din șirul lui Fibonacci.

9. Să se scrie o funcție recursivă pentru descompunerea în factori primi a unui număr natural.

10. Să se scrie o funcție recursivă pentru calculul valorii lui 2 la puterea n.

11. Scrieți o funcție recursivă pentru calculul combinărilor de n luate câte k.

12. Să se calculeze, folosind funcții recursive, sumele:

$$\begin{aligned} S_1 &= 1 + 1/2 + 1/3 + \cdots + 1/n \\ S_2 &= 1/2 + 2/1 + 2/3 + 3/2 + \cdots + n/(n+1) + (n+1)/n \end{aligned}$$

13. Scrieți o funcție recursivă care primește ca parametru un număr natural n și returnează numărul obținut din n prin eliminarea cifrelor pare.

    De exemplu, pentru $n = 134467$, funcția returnează 137.

14. Se consideră un număr natural care conține cel mult 9 cifre și se pune problema determinării numărului de cifre distincte ale numărului. Rezolvați problema folosind una sau mai multe funcții recursive.

    De exemplu, pentru $n = 41242$, se obține 3 (cifrele distincte sunt $4, 1, 2$).

1.6 Algoritmi de sortare. Algoritmi de căutare

Tematică

- Sortare prin inserție

- Sortare prin selecție
- Metoda BubbleSort (sortarea prin interschimbare sau metoda bulelor)
- Sortare prin numărare
- Tehnica interclasării
- Metoda QuickSort (sortarea rapidă folosind tehnica "Divide et impera")
- Căutare secvențială
- Căutare binară

Sumar teoretic

Considerăm o secvență de "entități" (numere, structuri conținând mai multe informații etc.). Fiecare entitate posedă o caracteristică numită cheie de sortare. Valorile posibile ale cheii de sortare aparțin unei mulțimi pe care există o relație totală de ordine.

Sortarea secvenței presupune aranjarea elementelor astfel încât valorile cheii de sortare să fie în ordine crescătoare (sau descrescătoare).

Proprietăți ale metodelor de sortare:

1. `Eficiența:` Metoda de sortare ar trebui să necesite un volum rezonabil de resurse (timp de execuție).
2. `Stabilitate:` O metodă de sortare este stabilă dacă păstrează ordinea relativă a elementelor având aceeași valoare a cheii de sortare.
3. `Simplitate:` Metoda de sortare ar trebui să fie simplu de înțeles și de implementat.
4. `Naturalețe:` O metodă de sortare este considerată naturală dacă numărul de operații necesare este proporțional cu gradul de "dezordine" al secvenței inițiale (care poate fi măsurat prin numărul de inversiuni din permutarea corespunzătoare secvenței sortate).

Câțiva dintre algoritmii de sortare[2], respectiv de căutare[3] sunt descriși în cele ce urmează:

Sortare prin inserţie

Ideea de bază a acestei metode este: *Începând cu al doilea element al tabloului* $x[1..n]$, *fiecare element este inserat pe poziţia adecvată în subtabloul care îl precede.*

La fiecare etapă a sortării, elementului $x[i]$ i se caută poziţia adecvată în subtabloul destinaţie $x[1..i-1]$ (care este deja ordonat) comparând pe $x[i]$ cu elementele din

[2]Daniela Zaharie, *Introducere in proiectarea si analiza algoritmilor,* Editura Eubeea, 2008.

[3]Steven S. Skiena, *The Algorithm Design Manual (Texts in Computer Science),* Third edition, Springer, 2020.

$x[1..i-1]$ începând cu $x[i-1]$ şi creând spaţiu prin deplasarea spre dreapta a elementelor mai mari decât $x[i]$.

Structura generală este descrisă după cum urmează:

```
insertie(real x[1..n])
for i ← 2,n do
     insereaza x[i] in subsirul x[1..i-1] astfel incat x[1..i] sa fie
        ordonat
end for
```

Algoritmul detaliat este următorul:

```
insertie(real x[1..n])
for i ← 2,n do
   j ← i-1
   aux ← x[i]
   while j>=1 and aux < x[j] do
      x[j+1] ← x[j]
      j ← j-1
   end while
   x[j+1] ← aux
end for
return (x[1..n])
```

Sortare prin selecţie

Ideea de bază a acestei metode este: *Pentru fiecare poziţie i, începând cu prima, se selectează din subtabloul ce începe cu acea poziţie cel mai mic element şi se amplasează pe locul respectiv (prin interschimbare cu elementul curent de pe poziţia i).*

Structura generală este descrisă după cum urmează:

```
selectie(real x[1..n])
for i ← 1,n-1 do
 se determina valoarea minima din x[i..n] si se interschimba cu x[i]
end for
```

Algoritmul detaliat este următorul:

```
selectie(real x[1..n])
for i ← 1,n-1 do
  k ← i
  for j← i+1,n do
     if x[k]>x[j] then
        k ← j
     end if
  end for
  if k != i then
     x[k] ↔ x[j]
  endif
```

```
end for
return (x[1..n])
```

Sortare prin interschimbare

Idee de bază: Tabloul este parcurs de la stânga spre dreapta și elementele adiacente sunt comparate. Dacă nu sunt in ordinea dorită atunci se interschimbă. Procesul este repetat până când tabloul ajunge să fie ordonat.

Structura generală este descrisă după cum urmează:

```
for i ← n,2,-1 do
 se parcurge x[1..i-1], se compara elementele adiacente si se interschimba
    daca este cazul
end for
```

Algoritmul detaliat este următorul:

```
Bubblesort(x[1..n])
for i ← n,2, -1 do
  for j← 1,i-1 do
     if x[j]>x[j+1]
      then x[j] ↔ x[j+1]
     end if
  end for
end for
return x[1..n]
```

Sortare prin interclasare

Idee de bază:

- Se împarte x[1..n] în două subtablouri x[1..[n/2]] and x[[n/2]+1..n];
- Se sortează fiecare subtablou;
- Se interclasează elementele subtablourilor x[1..[n/2]] și x[[n/2]+1..n] și se construiește tabloul sortat t[1..n]. Transferă conținutul tabloului temporar t în x[1..n].

Observații:

- Valoarea critică: 1 (un tablou conținând un singur element este implicit sortat);
- Valoarea critică poate fi mai mare decât 1 (de exemplu, 10), iar sortarea subtablourilor cu un număr de elemente mai mic decât valoarea critică se poate realiza cu unul dintre algoritmii elementari (de exemplu, sortare prin inserție).

Structura generală este descrisă după cum urmează:

```
sortare(x[s..d])
if s<d then
   m ← (s+d) DIV 2              //divizare
   x[s..m] ← sortare(x[s..m])   //rezolvare
   x[m+1..d]← sortare(x[m+1..d])
   x[s..d] ← interclasare(x[s..m], x[m+1..d]) //combinare
end if
return x[s..d]
```

Observație: Algoritmul se apelează prin `sortare(x[1..n])`.

Algoritmul detaliat este următorul:

```
interclasare (x[s..m], x[m+1..d])
i ← s; j ← m+1; k ← 0;
//se parcurg subtablourile in paralel si, la fiecare pas, se transfera cel
    mai mic element
while i <= m and j <= d do
   k ← k+1
   if x[i] <= x[j]
      then
         t[k] ← x[i]
         i ← i+1
      else
         t[k] ← x[j]
         j ← j+1
   end if
ens while
//se transfera eventualele elemente ramase in primul subtablou
while i <= m do
   k ← k+1
   t[k] ← x[i]
   i ← i+1
end while
//se transfera eventualele elemente ramase in al doilea subtablou
while j <= d do
   k ← k+1
   t[k] ← x[j]
   j ← j+1
end while
return t[1..k]
```

Interclasarea este o prelucrare ce poate fi utilizată pentru construirea unui tablou sortat pornind de la alte două tablouri sortate $(a[1..p], b[1..q])$.

Varianta de interclasare bazată pe valori santinelă: se adaugă două valori mai mari decât elementele tablourilor $a[p+1] = \infty, b[q+1] = \infty$.

```
interclasare(a[1..p], b[1..q])
a[p+1] ← \infty; b[q+1] ← \infty
    i ← 1; j ← 1;
```

```
    for k ← 1, p+q do
       if a[i] <= b[j]
          then c[k] ← a[i]
             i ← i+1
          else c[k] ← b[j]
             j ← j+1
       end if
    end for
return c[1..p+q]
```

Observații:

1. Principalul dezavantaj al sortării prin interclasare este faptul că utilizează un tablou adițional de dimensiunea tabloului de sortat.
2. Dacă în pasul de interclasare se folosește inegalitatea de tip "⇐" atunci sortarea prin interclasare este stabilă.

Căutare binară

Să se verifice dacă o valoare dată, v, aparține sau nu unui tablou ordonat crescător, $x[1..n]$ $(x[i] <= x[i+1], i = 1..(n-1))$.

Idee: se compară v cu elementul din mijloc și se continuă căutarea fie în subtabloul stâng, fie în cel drept.

Varianta recursivă:

```
cautbin(x[s..d],v)
if s>d then return False
else
    m ←(s+d) DIV 2
    if v==x[m] then return True
    else
       if v<x[m]
         then return cautbin(x[s..m-1],v)
         else return cautbin(x[m+1..d],v)
       end if
    end if
end if
```

Apelul funcției:

```
cautbin(x[1..n],v)
```

(la inceput $s = 1, d = n$).

Varianta iterativă 1:

```
cautbin1(x[1..n],v)
  s ← 1
  d ← n
```

```
  while s<=d do
     m ← (s+d) DIV 2
     if v==x[m] then return True
     else
        if v<x[m]
            then d ← m-1
            else s ← m+1
        end if
     end if
  end while
return False
```

Varianta iterativă 2:

```
cautbin2(x[1..n],v)
  s ← 1
  d ← n
  while s<d do
     m ←(s+d) DIV 2
     if v<=x[m]
        then d ← m
        else s ← m+1
     end if
  end while
  if x[s]==v then return True
             else return False
  end if
```

Probleme

1. Să se scrie un program care ordonează crescător elementele din prima jumătate a unui vector și descrescător elementele din a doua jumătate. Numărul de elemente ale vectorului este par.

   Exemplu: Vectorul 8 2 9 4 5 7 după sortare devine 2 8 9 7 5 4.

2. Se consideră un set de obiecte caracterizate prin dimensiune şi culoare (sunt trei culori posibile: roşu, alb, albastru). Să se ordoneze obiectele după culoare (prima dată obiectele roşii, apoi cele albe şi la final cele albastre) astfel încât obiectele cu aceeaşi culoare să fie ordonate crescător după dimensiune.

   Exemplu

   > `Intrare:` (4, alb), (3, roșu), (6, roșu), (5, albastru), (2, alb), (2, roșu), (4, albastru), (1, roșu)
   >
   > `Ieșire:` (1, roșu), (2, roșu), (3, roșu), (6, roșu), (2, alb), (4, alb), (4, albastru), (5, albastru)

   Indicație: Culoarea este reprezentată ca șir de caractere.

3. Se dau înălțimile a n copii, numerotați de la 1 la n, exprimate prin numere naturale. Afișați numerele de ordine ale copiilor în ordinea crescătoare a înălțimii lor.

   Exemplu

   `Intrare:` 80, 160, 120, 110, 90, 70, 100

   `Ieșire:` 6, 1, 5, 7, 4, 3, 2

4. Adaptați algoritmul de sortare prin numărare pentru ordonarea descrescătoare a unui tablou.

5. Scrieţi o funcţie/procedură care transformă un tablou unidimensional cu elemente numere reale, primit ca parametru, astfel încât elementele aflate pe poziţii pare să fie ordonate descrescător, iar cele aflate pe poziţii impare să fie ordonate crescător (poziţiile sunt contorizate începând cu 0, primul element este pe poziţie pară, al doilea pe poziţie impară etc.). De exemplu, tabloul $[5, 1, 7, 6, 2, 1, 8, 3]$ se transformă în $[8, 1, 7, 1, 5, 3, 2, 6]$.

6. Se dă un vector x cu n elemente numere naturale, ordonate crescător, și un vector y cu m elemente, de asemenea numere naturale. Verificați pentru fiecare element al vectorului y dacă apare în x.

   Indicație: Folosiți căutarea binară și căutarea secvențială.

7. Se dau n numere naturale distincte. Determinaţi câte triunghiuri distincte pot avea lungimile laturilor printre aceste numere.

   Exemplu

   `Intrare:` 5 (numărul de elemente) și 3 5 10 7 6 (elementele tabloului),

   `Ieșire:` 7

8. Se consideră două mulțimi A și B reprezentate prin tablouri având elementele specificate în ordine strict crescătoare. Să se construiască tablourile ce conțin reuniunea și intersecția lui A și B.

   Indicație: Folosiți tehnica interclasării.

9. Se dau două şiruri, cu n, respectiv m elemente, numere naturale. Primul şir este ordonat crescător, iar al doilea este ordonat descrescător. Să se afişeze, în ordine crescătoare, valorile pare din cele două şiruri.

   Exemplu

   `Intrare:` 5 (numărul de elemente ale șirului 1), 2 4 7 37 42 (elementele șirului 1)

   8 (numărul de elemente ale șirului 2), 88 88 67 45 42 32 4 1 (elementele șirului 2)

   `Ieșire:` 2 4 4 32 42 42 88 88

   Indicație: Folosiți tehnica interclasării.

10. Se dă un număr natural x și două tablouri a și b, cu n, respectiv m elemente, numere naturale, ordonate strict crescător. Să se afișeze, în ordine crescătoare, multiplii lui x care se află doar în unul dintre cele două șiruri.

    Indicație: Se parcurg cele două tablouri în paralel (în manieră balansată).

    - Dacă $a[i] = b[j]$ atunci se ignoră și se progresează în ambele tablouri,
    - Altfel se verifică dacă valoarea mai mică (dintre $a[i]$ și $b[j]$) este multiplu al lui x (dacă este, se reține valoarea) și se progresează în tabloul care conține valoarea mai mică.

11. La secția de împachetare a produselor dintr-o fabrică lucrează n muncitori. Fiecare muncitor împachetează același tip de produs și, pentru fiecare, se cunoaște timpul necesar pentru împachetarea unui obiect. Să se determine durata minimă de timp în care vor împacheta cei n muncitori cel puțin M obiecte.

    Exemplu

    `Intrare:` 6 (număr muncitori), 60 (număr obiecte), 4 7 3 6 7 1 (durate)

    `Ieșire:` 30

    Indicație: Se simulează câte un "cronometru" pentru fiecare muncitor utilizând un vector ale cărui elemente se incrementează până ajung la valoarea corespunzătoare timpului necesar împachetării unui obiect (moment în care se marchează faptul că s-a finalizat împachetarea unui obiect și se resetează "cronometrul").

12. Fiind dat un tablou de numere întregi nesortate, transformați tabloul astfel încât să fie sortat în formă de "undă". Un tablou $arr[0...n-1]$ este sortat în formă de undă dacă $arr[0] \geq arr[1] \leq arr[2] \geq arr[3] \leq arr[4] \geq ....$

    Exemplu

    `Intrare:` $arr[] = \{10, 5, 6, 3, 2, 20, 100, 80\}$

    `Ieșire:` $arr[] = \{10, 5, 6, 2, 20, 3, 100, 80\}$ sau $\{20, 5, 10, 2, 80, 6, 100, 3\}$ sau alt tablou care respectă această formă

    `Intrare:` $arr[] = \{20, 10, 8, 6, 4, 2\}$

    `Ieșire:` $arr[] = \{20, 8, 10, 4, 6, 2\}$ sau $\{10, 8, 20, 2, 6, 4\}$ sau alt tablou care respectă această formă

    Indicație: Se ordonează crescător tabloul, după care se interschimbă valorile elementelor vecine (primul cu al doilea, al treilea cu al patrulea, al cincilea cu al șaselea etc.).

13. Fiind dată o mulțime de intervale, verificați dacă se suprapun două dintre acestea.

    Exemplu

    `Intrare:` $arr[] = \{[1, 3], [5, 7], [2, 4], [6, 8]\}$

    `Ieșire:` Adevărat. Intervalele $[1, 3]$ și $[2, 4]$ se suprapun

`Intrare:` $arr[] = \{[1, 3], [7, 9], [4, 6], [10, 13]\}$

`Ieșire:` Fals. Nu există nicio pereche de intervale care să se suprapună

Indicație: Se ordonează intervalele crescător după momentul de start și se compară limita superioară a unui interval cu limita inferioară a intervalului următor. Dacă prima valoare este mai mare decât a doua, atunci există suprapunere.

14. Se consideră un vector de numere distincte inițial sortat în ordine crescătoare. Vectorul a fost rotit în sensul acelor de ceasornic de k ori. Pornind de la vectorul rotit, determinați valoarea lui k.

Exemplu:

`Intrare:` $v = \{15, 18, 2, 3, 6, 12\}$,

`Ieșire:` 2

Explicație: Vectorul inițial este $\{2, 3, 6, 12, 15, 18\}$. Din el se obține vectorul dat după 2 rotații.

Indicație: Se determină cel mai mic indice i cu proprietatea că valoarea elementului de pe poziția i este mai mare decât valoarea elementului de pe poziția $i + 1$.

1.7 Prelucrări asupra şirurilor de caractere

Tematică

Algoritmi de prelucrare a șirurilor de caractere:

- Separarea unui text în cuvinte.
- Analiza proprietăților unui șir de caractere (de exemplu, numărul de consoane/ vocale conținute, anagrame etc.).
- Verificarea satisfacerii unor reguli în construirea unui şir de caractere.
- Găsirea de șabloane în text.

Sumar teoretic

Caractere

O variabilă de tip caracter poate stoca un singur caracter. Tipul de date caracter este unul dintre tipurile implicite de date ale limbajelor de programare și ocupă de obicei un spațiu de memorie mic (de exemplu, limbajul C/C++ - 1 Byte). Un caracter este reținut intern sub forma unei valori întregi (cod ASCII).

Declararea și inițializarea unei variabile de tip caracter se realizează astfel:

- C/C++ `char c; char c1 = 'a';`

- PASCAL `var c : char;`

O valoare de tip caracter se specifică între apostroafe (de exemplu, `c='F';`).

Uneori este nevoie să obținem codul ASCII al unui caracter (valoarea returnată este un număr întreg), pentru această operație putem folosi:

- C/C++ conversia explicită de la tipul char la tipul întreg (de exemplu, `char c = 'k';` `int cod = (int) c;`)
- PASCAL funcția *ord*() (de exemplu, `var c : char; var x : integer; c := 'k';` `x := ord(x)`)

Transformarea în sens invers de la codul ASCII la caracter se realizează prin:

- C/C++ conversia explicită de la tipul întreg la tipul char (de exemplu, `int cod = 65;` `char c = (char) cod;`)
- PASCAL prin funcția *chr*() sau conversie de la tipul întreg la tipul char (de exemplu, `var c : char; var x : integer; x := 65; c := chr(x);`)

Șiruri de caractere

În limbajul C/C++ șirurile de caractere sunt tablouri unidimensionale de caractere care se termină cu un caracter special '\0' numit terminator de șir. Caracterul '\0' are codul ASCII 0, el nu se scrie și nu se citește.

Un șir de caractre se declară similar cu un tablou unidimensional de caractere (`char sir[100];`), doar că trebuie să alocăm o poziție în plus pentru caracterul terminator de șir. Dacă acesta lipsește din tabloul de caractere al șirului, tabloul nu este considerat a fi un șir de caractere, este un simplu tablou de caractere și funcțiile predefinite din biblioteca `string.h` nu vor funcționa corect.

Pentru a declara și inițializa un șir de caractere putem folosi:

(i) o construcție similară cu a tablourilor de caractere
`int s[]={'a', 'd', 'm', 'i', 't', 'e', 'r', 'e', '\0'};`

(ii) folosind un șir de caractere `int s[]="admitere";`. Spre deosebire de caracterele care sunt încadrate între apostroafe, șirurile de caractere sunt specificate între ghilimele, terminatorul de șir se adaugă implicit.

Șirurile de caractere sunt tablouri unidimensiuonale, iar accesul la elementele șirului se face prin intermediul operatorului de indexare [] (de exemplu, pentru a accesa litera 'm' a cuvântului "admitere" putem scrie `char c = s[3]`). Indexul literei 'm' în cuvântul "examen" este 3 deoarece prima literă a cuvântului are indexul 0 (indexul primului element al unui tablou în limbajul C/C++ este zero).

Limbajul C/C++ pune la dispoziție biblioteca de funcții `string.h` pentru a ușura lucrul cu șirurile de caractere. Printre funcțiile din biblioteca `string.h` se află:

- *strlen*(*sir*) - întoarce numărul de caractere vizibile din șirul de caractere (de exemplu, `strlen("admitere");` va întoarce valoarea 8);
- *strcmp*($sir1, sir2$) - compară lexicografic două șiruri de caractere, întorcând valoarea -1 dacă primul șir este mai mic decât al doilea, valoarea 0 dacă cele două șiruri sunt egale, valoarea 1 dacă primul șir este mai mare decât al doilea.
- *strcpy*($sir1, sir2$) copiază în variabila $sir1$ conținutul variabilei $sir2$; pentru ca operația să se realizeze logic corect zona de memorie alocată pentru variabila $sir1$ trebuie să fie cel puțin egală cu lungimea șirului $sir2$ (`strlen(șir2)+1`)
- *strncpy*($sir1, sir2, nrCaractere$) copiază în variabila $sir1$ $nrCaractere$ din conținutul variabilei $sir2$;
- *strcat*($sir1, sir2$) concatenează la conținutul șirului $sir1$, conținutul șirului $sir2$; pentru ca operația să se realizeze logic corect zona de memorie alocată pentru variabila $sir1$ trebuie să fie cel puțin egală cu lungimea șirului $sir1$ + lungimea $sir2$ + 1 (+1 pentru terminatorul de șir).

În limbajul PASCAL șirurile de caractere pot fi declarate în diferite moduri:

- tablou de caractere (de exemplu, `var sir: array[1..10] of char;`)
- variabila de tip string (de exemplu, `var sir: string; var sir1: string[10];`)

Spre deosebire de limbajul C/C++, în limbajul PASCAL șirurile caractere se specifică între apostroafe (ex. `sir := 'examen'`), indexul primului caracter din șir este 1, deci pentru a accesa litera 'm' va trebui să utilizăm indexul 4 (`var c: char; c := sir[4];`). În limbajul PASCAL, dacă accesăm indexul 0, vom obține lungimea șirului de caractere, reprezentarea internă fiind diferită de cea din limbajul C/C++.

În limbajul PASCAL se pot folosi operatori relaționali (<, <=, =, >, >=) pentru a compara două șiruri de caractere și operatrul + pentru a concatena două șiruri de caractere. La fel ca și în limbajul C/C++ există o serie de funcții utile pentru lucrul cu șiruri de caractere:

- *length*(*sir*) întoarce numărul de caractere din șirul de caractere;
- $sir3 := concat(sir1, sir2)$ întoarce un nou șir concatenat din conținutul variabilelor $sir1$ și $sir2$;
- $sir := copy(sursa, start_{index}, nr\_caractere)$ întoarce un nou șir care reprezintă o copie a șirului $sursa$;
- $delete(sir, start_{index}, nr\_caractere)$ ștergerea unui subșir.

Observație: `Problemele` se vor rezolva fără a folosi funcții predefinite de prelucrare a șirurilor de caractere.

Probleme

1. Se dă un text de maxim 256 de caractere. Să se afle următoarele informații despre textul dat.

   (a) Câte propoziții conține textul? Separatorii de propoziție se consideră caracterele: “”, “!” și “?”.

   Obiectiv: Identificarea propozițiilor din text

   Indicație: Se va defini o funcție care primește ca parametru textul și va întoarce numărul de propoziții din text.

   În cadrul funcției se declară un tablou cu separatorii de propoziție.

   Apoi se parcurge textul caracter cu caracter și se verifică dacă caracterul curent se află în tabloul de separatori. Dacă da se va incrementa o variabilă care reține numărul de propoziții din text.

   Observație: Se presupune că nu există 2 separatori de propoziție unul după altul de exemplu textul “Of!!!!!” nu este valid.

   (b) Pentru fiecare dintre propozițiile din text să se determine numărul de cuvinte. Cuvintele unei propoziții sunt separate printr-un singur spațiu. Scrieți o funcție care întoarce cea mai lungă propoziție din text (lungimea unei propoziții este definită ca fiind numărul de cuvinte pe care le conține).

   Obiectiv: Identificarea cuvintelor din text, calcul maximului dintr-o secvență de valori

   Indicație: Se va defini o funcție care primește ca parametru textul și va întoarce numărul maxim de cuvinte dintr-o propoziție.

   În cadrul funcției se declară un tablou cu separatorii de propoziție.

   Apoi se parcurge textul caracter cu caracter și se verifică dacă caracterul curent este spațiu. Dacă da, se va incrementa o variabilă care reține numărul de cuvinte din text.

   Observație: Atenție la ultimul cuvânt din text, dacă după el nu este spațiu nu se va număra.

   (c) Scrieți o funcție care returnează numărul de vocale și consoane din textul dat.

   Obiectiv: Parcurgerea unui text, numărarea caracterelor care aparțin unei mulțimi

   Indicație: Se va defini o funcție care primește ca parametri textul și două referințe în care se vor stoca (întoarce) numărul de consoane și numărul de vocale.

   În cadrul funcției se declară un tablou care conține vocalele (litere mari și mici).

   Apoi se parcurge textul caracter cu caracter și se verifică dacă caracterul curent este în intervalul [‘a’ ... ‘z’] sau [‘A’ ... ‘Z’]. Dacă este literă, se verifică dacă aparține mulțimii de vocale şi, în caz afirmativ, se crește numărul de vocale, altfel se crește numărul de consoane.

2. *Anagrame (un cuvânt este anagrama altui cuvânt dacă este constituit din aceleași litere dar în altă ordine).* De exemplu, “arc” este anagramă a cuvântului “rac”.

(a) Să se scrie o funcție care să verifice dacă, într-o pereche de cuvinte, un cuvânt este anagrama celuilalt cuvânt.

Obiectiv: Verificarea proprietăților unui cuvânt

Indicație: Se va defini o funcție care primește ca parametri două cuvinte (șiruri de caractere) și întoarce o valoare care specifică dacă cuvintele sunt sau nu anagrame.

Dacă cuvintele nu au aceeași lungime ele nu sunt anagrame.

Dacă cuvintele au aceeași lungime:

- se construiesc vectori de frecvențe pentru fiecare cuvânt (de câte ori apare o literă în cuvânt);
- se verifică dacă vectorii de frecvențe sunt identici, iar în caz afirmativ cuvintele sunt anagrame.

Observație: O altă implementare se poate baza pe construirea mulțimii de litere pentru fiecare cuvânt în ordine crescătoare și apoi se verifică dacă cele două mulțimi sunt identice.

(b) Să se scrie o funcție care verifică dacă o mulțime de cuvinte este formată din anagrame ale aceluiași cuvânt. De exemplu, mulțimea "arc", "rac", "car" satisface proprietatea cerută.

Obiectiv: Verificarea proprietăților unei mulțimii

Indicație: Se va defini o funcție care primește ca parametri o mulțime de cuvinte (tablou de șiruri de caractere) și întoarce o valoare care specifică dacă toate cuvintele din mulțime sunt anagrame.

Se definește o variabilă care va stoca o valoare (o numim `esteAnagrama`) care reprezintă faptul că presupunem că proprietatea pe care vrem sa o verificăm este adevărată.

Se parcurge mulțimea de cuvinte începând cu al doilea element al mulțimii și se verifică dacă primul element al mulțimii și elementul curent sunt anagrame (folosind funcția definită mai sus). Dacă nu sunt, atunci valoarea variabilei `esteAnagrama` se modifică pentru a reflecta că proprietatea nu mai este satisfăcută.

(c) Să se scrie o funcție care afișează pentru o mulțime de cuvinte toate perechile de cuvinte anagrame.

Obiectiv: Generarea submulțimilor unei mulțimi care verifică o proprietate

Indicație: Se va defini o funcție care primește ca parametri o mulțime de cuvinte (tablou de șiruri de caractere) și va afișa submulțimile formate din cuvinte anagramate.

Pentru a rezolva problema putem să considerăm un tablou de apariții care să ne spună dacă cuvântul a fost sau nu folosit (afișat).

Se parcurge tabloul de cuvinte și, pentru fiecare cuvânt, se verifică dacă există un alt cuvânt care este anagrama acestuia (folosind funcția anterioară care verifică dacă două cuvinte sunt anagrame). Un cuvânt deja afişat este marcat (reţinut în tabloul de aparitii) .

3. Să se genereze pentru un cuvânt dat toate prefixele și sufixele acelui cuvânt.

Exemplu: pentru cuvântul "paranteza"

- prefixele sunt “p”, “pa”, “par”, “para”, “paran”, “parant”, “parante”, “parantez”

  Obiectiv: Generarea unor șiruri care satisfac anumite proprietăți

  Indicație: Problema este asemănătoare cu problema afișării elementelor unei matrici aflate sub diagonala principală

- Sufixele sunt “a”, “za”, “eza”, “teza”, “nteza”, “anteza”, “ranteza”, “aranteza”

  Indicație: Problema este asemănătoare cu problema afișării elementelor unei matrici aflate deasupra diagonalei principale

4. Se dă un tablou de cuvinte, format doar din litere mici. Afișați șirul de litere din care sunt formate cuvintele în ordinea descrescătoare a literelor. Exemplu: pentru mulțimea de cuvinte “ana”, “are”, “mere” se obține rrnmeeeaaa.

   Obiectiv: Folosire tablou de frecvențe

   Indicație: Se va defini o funcție care primește ca parametru un tablou de cuvinte (eventual lungimea tabloului).

   Se va declara un tablou în care să se poată reține frecvența de apariție a fiecărei litere în tabloul de cuvinte. Elementele tabloului vor fi inițializate cu 0.

   Se vor parcurge toate elementele tabloului si pentru fiecare element (cuvânt) se vor parcurge literele din care este format și se va incrementa în tabloul de frecvențe valoarea care corespunde indexului din tablou atașat literei. De exemplu dacă litera a apare de 3 ori în text, frecvența atașată literei a va fi 3.

   După construirea tabloului de frecvențe, acesta se va parcurge de la sfârșit spre început și, pentru fiecare valoare diferită de 0, se va crea un ciclu pentru a afișa litera de n ori.

5. Se dă un text și un cuvânt pe care vrem să îl căutăm în text. Propuneți funcții care realizează următoarele variante de căutări:

   (a) Identifică prima apariție a cuvântului în text și returnează indexul la care începe cuvântul căutat (nu folosiți în implementarea algoritmului funcții deja implementate în biblioteci care realizează același lucru). În cazul în care cuvântul nu este prezent în text va întoarce valoarea -1. De exemplu, pentru textul “Ana are mere” şi cuvântul “are” se va returna 4, iar pentru cuvântul “care” se va returna -1.

   Obiectiv: Găsirea unui subșir într-un subșir (găsirea primei apariții)

   Indicație: Se definește o funcție care are ca parametri două cuvinte (șiruri de caractere) și va întoarce indexul la care începe prima apariție a subșirului în șir (de exemplu pentru șirul “abcdabcghbc” subșirul “bc” se găsește pentru prima dată la poziția 1, în cazul în care indicii încep cu 0).

   Pentru rezolvare se poate folosi algoritmul forței brute care presupune parcurgerea secvențială a primului șir (caracter cu caracter) și verificarea faptului dacă nu cumva subșirul începe la acea poziție.

   (b) Identifică toate aparițiile cuvântului în text și returnează un tablou care conține indecșii la care începe cuvântul căutat.

   Obiectiv: Găsirea unui subșir într-un subșir (găsirea tuturor aparițiilor)

Indicație: De exemplu, pentru șirul "abcdabcghbc" subșirul "bc" se găsește pe pozițiile $[1, 5, 9]$.

Se repetă prelucrarea de la subpunctul anterior.

(c) Identifică ultima apariție a cuvântului în text și returnează indexul la care începe cuvântul căutat, în caz contrar va întoarce valoarea -1.

Obiectiv: Găsirea unui subșir într-un subșir (găsirea ultimei apariții)

Indicație: De exemplu, pentru șirul "abcdabcghbc" subșirul "bc" se găsește pe poziția 9.

Pentru rezolvare se poate folosi algoritmul forței brute care presupune parcurgerea secvențială a primului șir (caracter cu caracter) și verificarea faptului dacă nu cumva subșirul începe la acea poziție. În acest caz, pentru a face algoritmul puțin mai eficient, în unele cazuri, se poate porni cu verificarea de la sfârșitul șirului.

6. Elevii unei clase trebuie să trimită rezolvările temelor la informatică sub forma unei arhive zip care trebuie să respecte următorul șablon de denumire `clasaX_nume_premune_zi-luna-an.zip`. Temele care sunt trimise și nu respectă acest format de numire nu vor fi corectate.

   Exemplu:

   - Dacă elevul Ionescu Ion din clasa a noua trimite următorul fișier `clasa9_ionescu_ion_02-04-2018.zip`, tema lui va fi luată în considerare pentru corectare.
   - Dacă elevul Vasilescu Vasile din clasa a unsprezecea trimite următorul fișier `clasa_11_vasilescu_vasile_02-03_2018.zip`, tema lui nu va fi luată în considerare pentru corectare.

   Cerințe:

   (a) Definiți o funcție care permite profesorului să verifice dacă numele fișierului trimis este corect sau nu (validarea se va face doar din punctul de vedere al structurii denumirii).

   Obiectiv: Verificarea dacă un șir de caractere respectă un șablon (pattern)

   Indicație: Se va căuta, pentru început, dacă există șiruri de caractere precizate care trebuie să se regăsească pe anumite poziții. De exemplu, subșirul "clasa" trebuie să fie la începutul cuvântului, iar subșirul ".zip" la sfârșitul cuvântului. Apoi trebuie identificați separatorii, în exemplul curent "_" (liniuță de subliniere) și "-" (cratimă). Apoi se verifică dacă subșirurile dintre aceștia respectă regulile impuse de șablon.

   (b) Profesorul și-a construit un tablou cu nume de fișiere. Determinați câte fișiere cu formatul corect au fost trimise.

   Obiectiv: Verificarea îndeplinirii unei proprietăți a elementelor unui tablou

   Indicație: Se va defini o funcție care primește ca parametru tabloul de cuvinte, eventual și lungimea tabloului și întoarce numărul de cuvinte din șir care respectă șablonul definit mai sus.

   Obiectiv: Crearea unui nou șir de caractere pe baza proprietăților unui alt șir de caractere

Indicație: Se va parcurge tabloul de cuvinte și se va folosi funcția definită la punctul (a) pentru a verifica dacă fiecare element al tabloului respectă sau nu șablonul. Dacă îl respectă, atunci se va crește valoarea variabilei în care se reține numărul de elemente care respectă șablonul.

(c) Ajutați profesorul să creeze o statistică care să îi spună câte teme are de corectat la fiecare clasă pornind de la tablourile cu nume de fișiere corespunzătoare diferitelor clase (câte fișiere în formatul corect au fost trimise pentru fiecare clasă).

Obiectiv: Creare de tabele de frecvență, identificare chei ale tabelului de frecvență

Inidicație: Pentru a simplifica problema, vom presupune că clasele sunt de la 0 la 12 (și în școală există o singură clasă pentru fiecare an).

În acest caz, ne putem folosi de ideea de la tabloul de frecvențe pentru a număra câți elevi au submis fișiere în formatul corect, pentru fiecare clasă.

Observație: Încercați să rezolvați problema în cazul general, când nu știm câte clase de aceleași fel sunt în școală (de exemplu, există clasele $9A, 9B, 9C$ și clasele $10A, 10B$).

7. Se dorește reducerea numărului de caractere pe care este stocat un text (comprimarea textului) prin înlocuirea caracterelor succesive care se repetă cu caracterul care se repetă și numărul de repetări. De exemplu, textul "abbcdeeeeff" va deveni "ab2cde4f".

(a) Propuneți un algoritm de comprimare a unui text.

Obiectiv: Modificări asupra unui șir de caractere (inserare/ștergere)

Obiectiv: Crearea unui nou șir de caractere pe baza proprietăților unui alt șir de caractere

Indicație: Se definește o funcție care primește ca parametru un șir de caractere și returnează noul șir de caractere.

Se parcurge caracter cu caracter șirul inițial și dacă două litere succesive sunt identice, atunci se mărește un contor în care se stochează numărul de litere identice găsite până în momentul curent, iar dacă două litere succesive sunt diferite, atunci se adaugă, în noul șir, litera. Dacă numărul de apariții este mai mare ca 1 se adaugă și el, apoi se resetează contorul la 1 deoarece s-a găsit o nouă literă.

(b) Pornind de la un text comprimat, propuneți un algoritm prin care obțineți textul inițial (de exemplu, "ab2cde4f2" devine "abbcdeeeeff").

Obiectiv: Generarea unui șir de caractere care respectă anumite proprietăți

Indicație: Se definește o funcție care primește ca parametru un șir de caractere și returnează noul șir de caractere.

Obiectiv: Înlocuirea unui subșir de caractere cu alt subșir de caractere într-un șir

Indicație: Se definește o funcție care primește ca argumente două șiruri de caractere (unul reprezintă cuvântul inițial și celălalt cuvântul în păsărească) și întoarce o valoare de adevăr care specifică dacă cuvântul inițial a fost corect transformat în păsărească.

Se parcurge, caracter cu caracter, șirul inițial și dacă caracterul e literă, atunci se adaugă la noul șir, iar dacă e cifră, atunci se transformă în număr pentru a afla numărul de repetări al literei. După ce am aflat numărul de repetări ale literei, adăugam la noul șir litera, de numărul de ori identificat.

Observație: Numărul de caractere care se repetaa poate fi mai mare de 9, deci rezultă că cifrele sunt formate din două sau mai multe caractere.

8. Elevii unei clase și-au propus să învețe limba păsărească astfel încât au nevoie de un program care să verifice dacă, pornind de la un cuvânt, au identificat corect cuvântul în limba păsărească. Scrieți o funcție care verifică dacă cuvântul din păsărească este corect identificat/codificat. Funcția primește ca parametri cuvântul în limba păsărească și cuvântul inițial și întoarce o valoare de adevăr în funcție de codificare. Codificarea în limba păsărească presupune înlocuirea fiecărei vocale astfel: "v" devine "vpv" şi păstrarea nemodificată a consoanelor (de exemplu, "a" devine "apa", iar "e" devine "epe"). De exemplu, cuvântul "capre", în păsărească, este "capaprepe".

   Indicație: Se va codifica cuvântul inițial în păsărească, apoi se vor verifica dacă cele două cuvinte sunt identice.

9. Să se verifice că un cuvânt este palindrom (citit de la dreapta la stânga și de la stânga la dreapta este același cuvânt). De exemplu, cuvântul "caiac" este palindrom, iar cuvântul "cor" nu este.

   (a) Propuneți o variantă iterativă de rezolvare a problemei.

   Obiectiv: Verificarea proprietăților unui șir de caractere, varianta iterativă

   Indicație: Se definește o funcție care primește ca argument un șir de caractere și întoarce o valoare de adevăr care spune dacă funcția este sau nu palindrom.

   Se pornește cu doi iteratori: unul de la sfârșitul șirului și unul de la începutul șirului. Atâta timp cât caracterele de pe pozițiile indicate de cei doi iteratori sunt egale, se descrește iteratorul care pornește de la sfârșitul șirului și se crește iteratorul de la începutul șirului, iar dacă cei doi iteratori ajung să se depășească înseamnă că este palindrom, altfel nu.

   (b) Propuneți o variantă recursivă de rezolvare a problemei.

   Obiectiv: Verificarea proprietăților unui șir de caractere, varianta recursivă

10. Ionel scrie poezii, dar face mici greșeli de tehnoredactare. Vom încerca să îl ajutăm pe Ionel să corecteze greșelile le tehnoredactare. Dacă versurile poeziei sunt stocate într-un tablou de șiruri de caractere (fiecare element din tablou conține un vers), trebuie să corectăm următoarele probleme:

    Obiectiv: Adăugarea/ștergerea de caractere dintr-un șir de caractere, generarea de sufixe

    (a) Deoarece dorim să folosim cât mai puține caractere pentru a stoca poezia, va trebui să verificăm dacă există caractere albe (spații și/sau taburi) la începutul și sfârșitul fiecărui vers și să le eliminăm.

    Indicație: Se definește o funcție care primește ca argument poezia (un tablou de șiruri de caractere), eventual numărul de versuri (dimensiunea tabloului).

Se va parcurge fiecare vers al poeziei și pentru fiecare vers:

- se verifică dacă ultimele caractere ale versului sunt caractere albe și se șterg prin deplasarea terminatorului de șir;
- se verifică dacă primele caractere sunt caractere albe: dacă da, atunci se deplasează versul cu un caracter la stânga.

(b) Din greșeală, Ionel a adăugat mai multe caractere albe (spații și taburi) între două cuvinte, acestea trebuie înlăturate (de exemplu, “ O, ce păcat , o, ce păcat”, “Că n-a fost fotografiat”, “ acel moment înălțător”, “Când, singur cuc, Apolodor”, “ a găurit cu un topor”, “ Pereții marelui vapor!...” se transformă în “O, ce păcat , o, ce păcat”, “Că n-a fost fotografiat”, “Acel moment înălțător”, “Când, singur cuc, Apolodor”, ”A găurit cu un topor”, “Pereții marelui vapor!...”.

Indicație: Se definește o funcție care primește ca argument poezia (un tablou de șiruri de caractere), eventual numărul de versuri (dimensiunea tabloului).

Se va parcurge fiecare vers al poeziei și, pentru fiecare vers, se verifică, caracter cu caracter, dacă două caractere succesive sunt caractere albe: dacă da, atunci se deplasează restul versului cu un caracter la stânga.

(c) Fiecare vers trebuie să înceapă cu literă mare de tipar.

Indicație: Se definește o funcție care primește ca argument poezia (un tablou de șiruri de caractere), eventual numărul de versuri (dimensiunea tabloului).

Se va parcurge fiecare vers al poeziei și, pentru fiecare vers, se verifică dacă primul caracter este literă mică: dacă da, atunci se transformă în literă mare (de exemplu, prin scăderea numărului de caractere dintre literele mici și literele mari).

(d) O altă problemă este legată de faptul că uneori înainte de caracterele ’,’, ’?’, ’!’ sau ’.’ s-a lăsat un spațiu liber și acesta trebuie eliminat sau s-a uitat să se lase spațiu după virgulă, punct, semnul exclamării și semnul întrebării. De exemplu, “Când , singur cuc,Apolodor” trebuie înlocuit cu “Când, singur cuc, Apolodor”.

Indicație: Se definește o funcție care primește ca argument poezia (un tablou de șiruri de caractere), eventual numărul de versuri (dimensiunea tabloului).

Se va parcurge fiecare vers al poeziei și, pentru fiecare vers, se verifică dacă caracterul curent este unul din caracterele: .,?!, iar dacă da:

- Se verifică dacă înaintea lui este spațiu: dacă da, atunci se deplasează șirul rămas cu o poziție la dreapta;
- Se verifică dacă după el este spațiu: dacă nu, atunci se deplasează șirul rămas cu o poziție la dreapta și se adaugă spațiul (**Atenție:** dacă este spațiu trebuie să deplasăm la dreapta; după ultimul caracter nu trebuie să adăugăm spațiu).

(e) Verificați tipul de rimă al poeziei: *împerecheată (aabb), încrucișată (abab), îmbrățișată (abba), monorima (aaaa)* sau *albă*, în restul cazurilor. Spunem că două cuvinte “rimează” dacă sufixele începând de la ultima vocală sunt identice, sufixele trebuie să aibă lungimea strict mai mare ca 1.

Indicație: Se definește o funcție care primește ca argument poezia (un tablou de șiruri de caractere), din poezie luăm în considerare doar primele patru versuri (pe baza lor determinăm tipul de rimă).

Pentru fiecare vers din cele patru se reține sufixul.

După ce s-au obținut sufixele, se verifică tipul de rimă.

11. Determinați numărul de cuvinte "magice" dintr-o propoziție. Cuvintele propoziției sunt separate de spații. Un cuvânt $(L_1L_2L_3...L_{n-1}L_n)$ este "magic" dacă satisface următoarea proprietate $|L_i - L_{i+1}| = |L_{n-i} - L_{n-i-1}|$, pentru oricare $i = 1...n$. De exemplu, în propoziția *"Miss Arora teaches us malayalam bdwy"* sunt 4 astfel de cuvinte: *Arora, us, Malayalam, bdwy.*

    Indicație: Se definește o funcție care primește ca argument propoziția (un șir de caractere), se parcurg literele propoziției, iar când se întâlnește caracterul spațiu, se consideră că s-a găsit un cuvânt și se verifică proprietatea de cuvânt "magic".

1.8 Structuri de date pentru grafuri

Tematică

Concepte și modalități de reprezentare pentru grafuri neorientate, grafuri orientate, arbori

- Reprezentarea grafurilor neorientate, a grafurilor orientate şi a arborilor
- Algoritmi de traversare şi analiză a grafurilor şi arborilor
- Proprietăţi ale grafurilor
- Aplicaţii

Sumar teoretic (Elemente de combinatorică)

Tehnici de numărare

Regula sumei. Dacă o procedură se poate efectua în 2 feluri, pentru felul i sunt n_i variante şi niciuna din variantele de primul fel nu coincide cu vreo variantă din felul 2, atunci există $n_1 + n_2$ variante de a efectua procedura.

Regula produsului. Dacă o procedură poate fi descompusă într-o secvenţă de 2 proceduri astfel încât:

- prima se poate efectua în n_1 feluri,
- a doua se poate efectua în n_2 feluri,

atunci există $n_1 * n_2$ feluri de a efectua acea procedură.

Permutări, aranjamente, combinări.

Se consideră o mulţime S cu n elemente.

Numărul de **permutări** ale unei mulţimi cu n elemente este $n!$

Numărul total de **submulţimi ordonate** ale lui S cu k elemente se numesc **aranjamente de n luate câte** k.

Exemplu. Fie mulţimea $A = \{a, b, c, d, e\}$. Se pot construi 20 de mulţimi ordonate având câte două elemente fiecare:

$(a, b), (a, c), (a, d), (a, e),$
$(b, a), (b, c), (b, d), (b, e),$
$(c, a), (c, b), (c, d), (c, e),$
$(d, a), (d, b), (d, c), (d, e),$
$(e, a), (e, b), (e, c), (e, d).$

Formule uzuale:

Pentru $\forall n \in N^*, k \in N, n \geq k$

1. Formula de recurenţă:
   $A_n^k = (n - k + 1) * A_n^{k-1}$
2. Formula factorială a aranjamentelor:
   $A_n^k = \frac{n!}{(n-k)!}$

Proprietăţi:

$A_n^n = 1 * 2 * 3 * ... * n = n!$
$A_n^0 = 1$
$A_n^{n-1} = A_n^n$

Numărul total de **submulţimi** ale lui S cu k elemente se numesc **combinări de** n **elemente luate câte** k.

1. Formula de recurenţă
   $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k, 0 < k < n$
2. Formula factorială:
   $C_n^k = \frac{n!}{k!*(n-k)!}$

Proprietăţi:

$C_n^0 = 1 = C_n^n$
$C_n^k = C_n^{n-k}$, pentru $0 \geq k \geq n$.

Probleme

1. Se citeşte un graf neorientat din fişierul `muchii.txt` în care avem pe prima linie numărul `n` de noduri şi cel de muchii separate prin spaţiu, iar pe fiecare din liniile următoare avem nodurile unei muchii separate prin spaţiu. Se presupune că $n \leq 50$.

   (a) Pentru fiecare nod, afişaţi gradul şi muchiile incidente la acel nod.

   (b) Afişaţi numărul de noduri şi matricea de adiacenţă a grafului.

2. Se citesc două grafuri neorientate, unul cu $n \leq 100$ vârfuri şi `m` muchii, iar celălalt cu $k \leq 100$ vârfuri şi `r` muchii, ambele reprezentate prin lista de muchii.

   Să se determine dacă al doilea graf este subgraf al primului graf.

3. Se dă un graf neorientat cu $n \leq 100$ noduri, pentru care se cunoaşte matricea de adiacenţă `a`. Să se determine componentele conexe ale grafului şi apoi să se indice:

   (a) numărul de componente conexe

   (b) pentru fiecare componentă conexă:

   - nodurile care fac parte din componenta respectivă
   - dacă are sau nu un ciclu.

4. Se dă un graf neorientat cu $n \leq 50$ noduri, pentru care se cunoaşte matricea de adiacenţă `a`. Determinaţi numărul minim de culori cu care se pot colora nodurile grafului astfel încât oricare două noduri adiacente să aibe culori diferite. Afişaţi pentru fiecare culoare nodurile care se colorează cu ea.

5. Se citeşte un graf orientat din fişierul `muchii.txt` în care avem pe prima linie numărul `n` de noduri şi cel de muchii separate prin spaţiu, iar pe fiecare din liniile următoare avem nodurile unei muchii separate prin spaţiu. Se presupune că $n \leq 50$. Apoi se citeşte de la consolă un număr întreg `m` astfel încât $0 < m \leq n$.

   Să se afişeze:

   (a) Câte subgrafuri cu `m` noduri ale grafului citit sunt circuite elementare.

   (b) Numărul maxim de muchii al unui subgraf cu `m` noduri.

6. Se dă un graf orientat cu $n \leq 50$ noduri, care se citeşte dintr-un fişier text `muchii.txt` în care avem pe prima linie numărul `n` de noduri şi cel de muchii separate prin spaţiu, iar pe fiecare din liniile următoare avem nodurile unei muchii separate prin spaţiu. Știind că:

   - Un *drum hamiltonian* este un drum care vizitează fiecare nod al grafului exact o dată. Un *ciclu hamiltonian* este un ciclu care vizitează fiecare nod o singură dată (cu excepția nodului care este și primul și ultimul). Un graf care conține un ciclu hamiltonian se numește *graf hamiltonian.*
   - Un *drum eulerian* este un drum care traversează fiecare muchie exact o dată. Un *ciclu eulerian* este un ciclu care parcurge fiecare muchie exact o dată. Un graf care conține un ciclu eulerian este numit *graf eulerian.*

să se determine

(a) Dacă graful este hamiltonian. Dacă este, să se afişeze un circuit hamiltonian.

(b) Dacă graful este eulerian. Dacă este, să se afişeze un circuit eulerian.

7. Se dă un arbore binar cu $\leq$ `100` noduri reprezentat prin vectorii de descendenţi `S` şi `D`. Pentru fiecare nod `i`, `S[i]` este descendentul stâng al nodului `i`, şi `D[i]` este descendentul drept al nodului `i`.

   Dacă `S[i]` = `0` atunci nodul `i` nu are descendent stâng, iar dacă `D[i]` = `0` atunci nodul `i` nu are descendent drept. Afişaţi pe rânduri separate:

   - Frunzele arborelui
   - Nodurile cu un singur descendent direct
   - Nodurile cu 2 descendenţi direcţi

8. Se dă un arbore binar cu $\leq$ `100` noduri reprezentat prin vectorii `T` şi `P`:

   - `T[i]` este nodul tată al nodului `i`; dacă nodul `i` nu are nod tată atunci `T[i]` = `0`.
   - `P[i]` = -1 dacă nodul `i` este descendent stâng; `P[i]` = `1` dacă nodul `i` este descendent drept.

   Calculaţi şi afişaţi vectorii `S` şi `D` de descendenţi ai nodurilor grafului:

   - `S[i]` este descendentul stâng al nodului `i`, şi
     `D[i]` este descendentul drept al nodului `i`.
   - Dacă `S[i]` = `0` atunci nodul `i` nu are descendent stâng, iar dacă `D[i]` = `0` atunci nodul `i` nu are descendent drept.

9. Se citeşte un graf conex neorientat cu `n` $\leq$ `100` noduri şi `m` muchii etichetate prin costui pozitive. Graful se citeşte dintr-un fişier text `muchii.txt` în care avem pe prima linie numărul `n` de noduri şi numărul `m` de muchii separate prin spaţiu, iar pe fiecare din liniile următoare avem nodurile şi costul unei muchii separate prin spaţiu. Se presupune că în fişier, muchiile apar în ordinea crescătoare a costului lor.

   Să se determine un arbore de acoperire cu cost minim al grafului citit, şi să se afişeze muchiile acestuia.

Capitolul 2

Subiecte date la Concursul FMI

2.1 Ediția a III-a (2017)

1. Un număr natural nenul este considerat *binar simetric* dacă reprezentarea sa în baza 2 este un șir simetric (prima cifră binară coincide cu ultima, a doua cu penultima etc.). De exemplu, 27 este binar simetric întrucât șirul binar corespunzător 110011 este simetric.

   (a) Scrieți o funcție (C/C++/Pascal) care primește ca parametru de intrare un număr natural și returnează numărul de cifre binare (de exemplu, 27 are 5 cifre binare).

   (b) Scrieți o funcție (C/C++/Pascal) care primește ca parametru de intrare un număr natural n și o valoare k cuprinsă între 1 și numărul de cifre binare ale lui n și returnează cifra binară de ordin k (cifra binară de ordin 1 este cea mai puțin semnificativă).

   (c) Scrieți o funcție (C/C++/Pascal) care primește ca parametru de intrare un număr natural și returnează 1 dacă numărul este *binar simetric* și 0 în caz contrar. Pentru verificarea proprietății nu se vor folosi tablouri.

2. Se consideră un șir de numere întregi distincte, nu neapărat ordonat, și se pune problema descompunerii sale într-un numar *cât mai mic* de subșiruri crescătoare astfel încât fiecare element din șirul inițial să se afle exact într-un subșir, iar ordinea relativă dintre elementele șirului inițial să se conserve și în cazul subșirurilor. De exemplu, șirul inițial $(13, 4, 21, 16, 18, 5, 3)$ se poate descompune în subșirurile $\{(13, 21), (4, 16, 18), (5), (3)\}$ sau în subșirurile $\{(13, 16, 18), (4, 5), (21), (3)\}$.

   (a) Propuneți o metodă de construire a subșirurilor.

   (b) Scrieți programul (C/C++/Pascal) care implementează metoda propusă. Programul va citi de la tastatură șirul de numere și va afișa subșirurile determinate.

3. Se consideră o matrice pătratică de dimensiune $n \times n$ cu elemente numere reale și se pune problema reorganizării ei astfel încât la parcurgerea elementelor în spirală, pornind de la elementul din colțul din stânga sus, în sensul acelor de ceasornic, să se obțină un șir ordonat crescător.

(a) Scrieți o funcție/procedură (C/C++/Pascal) care construiește un tablou ordonat crescător ce conține toate cele n^2 elemente ale matricii.

(b) Scrieți o funcție/procedură (C/C++/Pascal) care plasează elementele din tabloul construit la etapa anterioară, sub forma unei spirale care începe din colțul Stânga-sus al matricii, în sensul acelor de ceasornic.

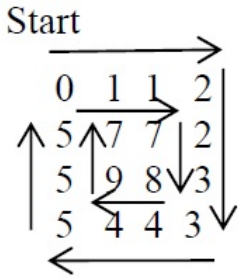
Exemplu: Intrare:

$$\begin{array}{cccc} 1 & 4 & 2 & 5 \\ 3 & 5 & 7 & 9 \\ 0 & 8 & 7 & 4 \\ 1 & 3 & 5 & 2 \end{array}$$

Rezultat:

(a) $0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 5, 7, 7, 8, 9$

(b)



4. Se consideră o succesiune de ($n-1$ celule care conțin operatori de inegalitate ("<", respectiv">") și se dorește să se plaseze *toate* valorile naturale $1, 2, \ldots, n$ între celule (inclusiv înaintea primeia și după ultima) astfel încât orice triplet de elemente consecutive < număr, operator, număr > să exprime relația corectă dintre cele două numere.

   Exemplu: Pentru succesiunea de operatori:

   $\boxed{>}$ $\boxed{<}$ $\boxed{<}$ $\boxed{>}$ $\boxed{>}$ $\boxed{<}$

   se poate obține (soluția nu este unică):

   $\boxed{2}$ $\boxed{>}$ $\boxed{1}$ $\boxed{<}$ $\boxed{3}$ $\boxed{<}$ $\boxed{6}$ $\boxed{>}$ $\boxed{5}$ $\boxed{>}$ $\boxed{4}$ $\boxed{<}$ $\boxed{7}$

   Scrieți un program (C/C++/Pascal) care citește de la tastatură secvența de operatori (ca șir de caractere) și afișează o secvență de valori și operatori intercalate care satisfac cerințele (pentru exemplul de mai sus se citește ">><<>><" și se afișează $2 > 1 < 3 < 6 > 5 > 4 < 7$).

Notă:

1. Pentru funcții/proceduri se vor specifica parametrii și se vor declara variabilele locale (dacă este cazul).
2. În cazul fiecărei probleme se va descrie ideea de rezolvare în limbaj natural sau se vor pune comentarii explicative în cadrul codului.

BAREM

Start ..... **10 pct.**

Problema 1 ..... **15 pct.**

(a) (definire funcție, declarații - 1p, calcul corect număr biți - 4p) ..... 5 pct

(b) (definire funcție, declarații - 1p, calcul corect cifra binară - 4p) ..... 5 pct

(c) (definire funcție, declarații - 1p, verificare simetrie - 4p) ..... 5 pct

Problema 2 ..... **25 pct.**

(a) (descrierea unei metode corecte de construire a subșirurilor) ..... 10 pct

(b) (declarații - 1p, citire șir - 1p, afișare subșiruri - 3p, construire subșiruri - 10p) ..... 15 pct

Problema 3 ..... **30 pct.**

(a) (definire funcție, declarații - 1p, parcurgere matrice - 2p, construire tablou ordonat - 7p) ..... 10 pct

(b) (construire matrice cu elemente plasate corect) ..... 20 pct

Problema 4 ..... **20 pct.**

(declarații - 1p, citire șir operatori - 1p, construirea unui șir de valori ce satisface restricțiile - 15p, afișare secvența de valori si operatori intercalate - 3p)

2.2 Ediția a IV-a (2018)

1. (30p)

   (a) (5p) Se consideră funcția $f : \mathbb{N}^* \times \mathbb{N}^* \to \mathbb{N}$ definită prin:

   $$f(m,n) = \begin{cases} \text{m} & \text{dacă } m < n \\ \text{f(m-n, n)} & \text{dacă } m \geq n \end{cases}$$

      i. Scrieți o funcție recursivă în C/Pascal care primește doi parametri m și n de tip întreg și returnează valoarea $f(m,n)$;

      ii. Ce returnează funcția atunci când este apelată pentru două valori naturale nenule?

   (b) (5p) Se știe că orice număr natural nenul poate fi scris, în mod unic, ca o sumă de termeni distincți care sunt puteri naturale ale 2. De exemplu, $9 = 2^3 + 2^0, 15 = 2^3 + 2^2 + 2^1 + 2^0, 32 = 2^5$. Scrieți o funcție care primește ca parametru un număr natural nenul, n, și returnează returnează numărul de termeni distincți din descompunerea lui n ca sumă de puteri ale lui 2.

   (c) (5p) Se consideră un număr natural mare (ce conține cel puțin 100 de cifre) specificat prin secvența cifrelor sale, $(c_k, c_{k-1}, \ldots, c_1, c_0)$ (cu c_k cifra cea mai semnificativă și c_0 cifra cea mai puțin semnificativă). Scrieți o funcție care primește ca parametri numărul de cifre și un tablou ce conține cifrele numărului și returnează 1 dacă numărul este divizibil cu 15 și 0 în caz contrar.

   (d) (5p) Se consideră o matrice pătratică A cu n linii și n coloane și se presupune că elementele aflate sub diagonala principală sunt stocate într-un tablou prin parcurgerea linie cu linie a matricii. Tabloul va conține elementele $a_{21}, a_{31}, a_{32}, a_{41}, a_{42}, a_{43} \ldots$. Presupunând că indicii tabloului pornesc de la 1, stabiliți care este indicele în tablou al elementului a_{ij} din matrice ($i > j$) :

      i. $(j-2)(j-1)/2 + i$;

      ii. $(i-2)(i-1)/2 + j$;

      iii. $(i-2)(j-1)/2 + i$;

      iv. $(j-2)(i-1)/2 + j$;

      v. niciuna dintre variante nu e corectă.

   (e) (5p) Coeficientul de "palindromitate" al unui șir de numere naturale $(b_1, b_2, \ldots, b_k)$ se definește ca fiind numărul minim de operații de incrementare care ar trebui aplicate unor elemente din șir, astfel încât șirul să devină palindrom ($b_1 = b_k, b_2 = b_{k-1}$ *etc.*).

      i. Scrieți relația matematică de calcul a coeficientului de "palindromitate";

      ii. Scrieți o funcție care primește ca parametru un tablou cu valori naturale și returnează valoarea coeficientului.

   (f) (5p) Se consideră un graf neorientat cu $n = 6$ noduri specificat prin lista muchiilor sale: $(1,2), (1,3), (2,3), (2,4), (3,6), (4,5), (4,6), (5,6)$.

      i. Deseneți graful.

ii. Este graful conex?

iii. Care este numărul minim de muchii care ar trebui eliminate pentru ca graful să devină conex? Dați un exemplu de muchii prin a căror eliminare graful devine neconex.

2. (20p) Se consideră un set de n ($10 \leq n \leq 100$) segmente de dreaptă specificate prin coordonatele extremităților lor care sunt stocate într-o matrice A cu n linii și 4 coloane având elemente numere reale. Fiecare linie din matrice conține coordonatele a două puncte în plan: a_{i_1} și a_{i_2} reprezintă abscisa, respectiv ordonata primei extremități, iar a_{i_3} și a_{i_4} reprezintă abscisa, respectiv ordonata celei de a doua extremități.

   (a) (5p) Scrieți o funcție/procedură care afișează numărul de segmente de dreaptă orizontale și numărul de segmente de dreaptă verticale.

   (b) (5p) Știind că lungimea unui segment determinat de două puncte având coordonatele (x_i, y_i), respectiv (x_j, y_j) este $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, scrieți o funcție care primește ca parametri coordonatele a două puncte și returnează lungimea segmentului determinat de ele.

   (c) (10p) Scrieți o funcție/procedură care transformă matricea A, astfel încât liniile sale să fie ordonate crescător după lungimea segmentului pe care îl reprezintă.

3. (20p) Pentru un număr natural n definim $d(n)$ ca fiind suma dintre n și suma cifrelor sale. De exemplu, $d(75) = 75 + 7 + 5 = 87$. Numărul n este considerat *generator* pentru $d(n)$ (75 este generator pentru 87). Un număr natural nenul m este considerat *auto-număr* dacă nu există niciun număr $n < m$ cu proprietatea că n este generator pentru m (adică $m \neq d(n)$ pentru orice $n < m$).

   (a) (5p) Scrieți o funcție care primește un număr natural n și returnează valoarea lui $d(n)$.

   (b) (10p) Scrieți o funcție care determină numărul maxim de generatori pe care îi poate avea un număr cu k cifre ($2 \leq k \leq 4$).

   (c) (5p) Scrieți o funcție/procedură care afișează toate auto-numerele care au cel mult k cifre ($2 \leq k \leq 4$).

4. (20p) Se consideră un set S de m șiruri de caractere ce conțin simboluri din mulțimea $\{a, b\}$ și se pune problema identificării unor subșiruri cât mai lungi care verifică șablonul "$a^n b a^n$"="$\underbrace{aa\ldots a}_{n \text{ ori}}\, b\, \underbrace{aa\ldots a}_{n \text{ ori}}$" ($n \geq 1$). De exemplu, în șirul "*abbaaabaaabaab*" cel mai lung subșir care respectă șablonul este "*aaabaaa*" (cu $n = 3$).

   (a) (10p) Scrieți o funcție care primește ca parametru un șir de caractere și returnează valoarea n corespunzătoare celui mai lung subșir care respectă șablonul. Dacă un astfel de subșir nu există, se returnează 0.

   (b) (10p) Declarați o variabilă în care se poate stoca setul de șiruri de caractere. Scrieți o funcție care afișează cel mai lung subșir care respectă șablonul și este comun tuturor șirurilor din setul S.

Notă:

1. Limbajul de programare este la alegere între C/C++ și Pascal.
2. Pentru fiecare dintre funcțiile/subprogramele scrise se vor specifica toate declarațiile de variabile globale și locale necesare și se vor pune comentarii explicative. **NU** este necesară scrierea programului principal și nici preluarea datelor prin citire.

BAREM

Start ..... **10 pct.**

Problema 1 ..... **30 pct.**

1a ..... 5 pct.

(*i.* descriere corectă a funcției recursive - 3p; *ii.* funcția determină restul împărțirii întregi a lui m la n - 2p)

1b ..... 5 pct.

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; contorizarea numărului de cifre egale cu 1 din reprezentarea în baza 2 a numărului - 4p)

1c ..... 5 pct.

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; verificare divizibilitate cu 3 (suma tuturor cifrelor este divizibilă cu 3) - 2p; verificare divizibilitate cu 5 (cifra cea mai puțin semnificativă este 0 sau 5) - 2p)

1d ..... 5 pct.

(*ii.* poziția elementului a_{ij} în tablou este $(i-2)(i-1)/2+j$ - 5p)

1e ..... 5 pct.

(*i.* $\sum_{i=1}^{[k/2]} |b_i - b_{k-i+1}|$ - 2p; *ii.* declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; calculul corect al coeficientului - 2p)

1f ..... 5 pct.

(*i.* desen corect al grafului - 1p; *ii.* da, graful este conex - 1p; *iii.* trebuie eliminate minim două muchii - 2p; exemplu corect de muchii eliminate (de exemplu, $(2,4)$ și $(3,6)$ - 1p)

Problema 2 ..... **20 pct.**

2a ..... 5 pct.

(declararea corectă a funcției, a variabilelor locale și afișarea rezultatelor - 1p; contorizarea corectă a segmentelor orizontale (segmentul i este orizontal dacă $a_{i_2} = a_{i_4}$, respectiv vertical dacă $a_{i_1} = a_{i_3}$) - 4p)

2b ..... 5 pct.

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; descrierea corectă a calculului lungimii unui segment - 4p)

2c ..... 10 pct.

(declararea corectă a funcției, a variabilelor locale/globale - 1p; implementarea corectă a algoritmului de sortare folosind funcția de la punctul (b) drept criteriu de sortare - 9p)

Problema 3 ..... **20 pct.**

3a ..... 5 pct.

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; calcul sumă cifre - 3p; calcul corect $d(n)$ - 1p)

3b ..... 10 pct.

(declararea corectă a funcției, a variabilelor locale și afișarea rezultatelor - 1p; construirea unui tabel de frecvențe corespunzător valorilor de tip $d(i)$ calculate pentru valori ale lui i cuprinse între 1 și $10^k - 1$ - 7p; determinarea valorii maxime din tabelul de frecvențe - 2p)

3c ..... 5 pct.

(declararea corectă a funcției, a variabilelor locale și afișarea rezultatelor - 1p; generarea corectă a auto-numerelor (de exemplu, parcurgând tabelul de frecvențe construit la punctul anterior) - 4p)

Problema 4 ..... **20 pct.**

4a ..... 10 pct.

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; analiza tuturor subșirurilor care respectă șablonul (de exemplu, pornind de la un subșir de forma "*aba*" și extinzându-l înspre stânga și dreapta cât este posibil) - 6p; determinarea corectă a parametrului n corespunzător șablonului de lungime maximă - 3p)

4b ..... 10 pct.

(declararea unei variabile care stochează setul de șiruri de caractere - 2p; declararea corectă a funcției și a variabilelor locale - 1p; determinarea pentru fiecare șir din set a lungimii maxime a unui subșir care respectă șablonul (folosind funcția de la punctul (a)) - 3p; determinarea valorii minime dintre lungimile maxime determinate - 2p; afișarea corectă a șablonului comun - 2p)

2.3 Ediția a V-a (2019)

1. (50p)

   (a) (10p) Se consideră două numere naturale nenule x și y. Să se construiască cel mai mare număr z care este constituit din cifrele distincte care apar atât în x cât și în y. De exemplu, pentru $x = 46106$ și $y = 12156$ se obține $z = 61$. Pentru determinarea lui z se vor scrie următoarele subprograme (funcții/proceduri):

      i. Un subprogram care primește ca parametru valorile x și y și construiește un tablou c cu 10 elemente având proprietatea: $c[i] = 1$ dacă i este cifră comună celor două numere și 0 în caz contrar.

      ii. Un subprogram care primește ca parametru tabloul c construit la punctul i. și returnează numărul z.

   (b) (10p) Se consideră o bucată dreptunghiulară de tablă de lungime m și lățime n (m și n sunt numere naturale) și se pune problema secționării ei exacte într-un număr cât mai mic de bucăți identice de formă pătrată.

      i. Descrieți pe scurt și argumentați modul în care poate fi calculată lungimea laturii pătratului și numărul minim de tăieturi necesare pentru a obține toate pătratele.

      ii. Scrieți un subprogram care primește ca parametri valorile m și n și afișează :

         A. numărul minim de pătrate;

         B. numărul minim de tăieturi.

   (c) (5p) Se consideră funcția $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ definită prin:

$$f(a,b) = \begin{cases} 0 & \text{dacă } b = 0 \\ \text{f(2a, [b/2])} & \text{dacă } b > 0, b \text{ este par} \\ \text{f(2a, [b/2])+a} & \text{dacă } b > 0, b \text{ este impar} \end{cases}$$

   unde $[x]$ reprezintă partea întreagă a numărului x.

      i. Care este valoarea funcției f pentru:

         A. $a = 3, b = 4$;

         B. $a = 4, b = 3$;

         C. a și b două numere naturale arbitrare.

      ii. Scrieți o funcție recursivă pentru calculul valorii $f(a, b)$ pentru două numere naturale a și b.

   (d) (5p) Se consideră o matrice A cu m linii și n coloane și elemente din $\{0, 1\}$ care descrie relația de prietenie pe o platformă socială ($A_{ij} = A_{ji} = 1$ dacă i este prieten cu j și 0 în caz contrar). Scrieți un subprogram (funcție) care primește ca parametru matricea și afișează persoana (persoanele) care au numărul maxim de prieteni.

   (e) (5p) Se pune problema acoperirii unei table de dimensiune $2 \times m$ folosind piese de dimensiune 1×2 sau 2×1. Fie $V(m)$ numărul de variante în care poate fi acoperită tabla de piese.

i. Care dintre următoarele afirmații este corectă?
   A. $V(m) = C_m^2$;
   B. $V(m) = m!$;
   C. $V(1) = 1, V(2) = 2, V(m) + V(m-1) + V(m-2)$, pentru $m > 2$;
   D. $V(1) = 1, V(m) = 2V(m-1)$, pentru $m > 1$.

ii. Argumentați alegerea făcută la punctul i.

(f) (15p) Se consideră un set de n puncte în plan date prin coordonatele lor carteziene (numere întregi) stocate în două tablouri x și y și se pune problema determinării numărului de linii drepte care trec prin originea sistemului de axe de coordonate și conțin cele n puncte.

i. Scrieți un subprogram (funcție/procedură) care ordonează tablourile x și y crescător după valoarea y/x (se presupune că toate valorile x sunt nenule). După ordonare, punctele vor fi în ordinea crescătoare a pantei dreptei determinată de origine și punctul respectiv.

ii. Scrieți o funcție care primește tablourile x și y ordonate și returnează numărul de linii distincte care trec prin origine și conțin toate cele n puncte.

2. (25p) Primarul orașului `WWW` dorește să instaleze o cameră de luat vederi în oraș. Se consideră că orașul are formă dreptunghiulară, cu străzi paralele și perpendiculare situate la distanțe egale. Dreptunghiul corespunzător orașului și străzile aferente sunt modelate printr-o grilă cu dimensiunea $M \times N (2 \leq M, N \leq 1000)$, în nodurile căreia sunt plasate clădirile (fiecare clădire este modelată printr-un "bețișor" de grosime neglijabilă a cărui înălțime este egală cu înălțimea clădirii din acel punct). Camera de luat vederi urmează să fie instalată la nivelul solului în colțul stânga-sus (NORD-VEST) al orașului (punctul de coordonate $(0, 0)$, unde nu se găsește nicio clădire) și se presupune că se poate roti complet în orice plan relativ la poziția în care a fost instalată.

O cameră de luat vederi poate supraveghea clădiri doar dacă le poate identifica parțial sau integral. De exemplu, considerând un oraș de dimensiune 3×3, în care înălțimile clădirilor sunt cele marcate în Figura 2.1, camera de luat vederi situată la nivelul solului va putea identifica pe direcția SUD ambele clădiri (de înălțime 1 și 3), pe direcția EST va putea identifica doar clădirea de înălțime 3, iar pe direcția SUD-EST va vedea doar clădirea de înălțime 2 situată pe diagonală. Un alt exemplu de clădiri vizibile pe o direcție dată este ilustrat în Figura 2.2 (clădirile marcate cu linie continuă sunt vizibile, cela marcate cu linie întreruptă sunt invizibile).

(a) (15p) Scrieți o funcție (denumită `viewSoutEast` care primește ca parametri o matrice O ce conține înălțimile clădirilor și returnează numărul de clădiri care pot fi supravegheate pe direcțiile EST, SUD și SUD-EST de către camera plasată în colțul $(0, 0)$ la nivelul solului.

(b) (5p) Se consideră o cameră de luat vederi care poate identifica clădiri într-o direcție arbitrară (definită de poziția camerei $(0, 0)$ și o poziție specificată (k_1, k_2) - corespunzătoare primei clădiri pe această direcție). Scrieți formula generală a poziției clădirilor aflate pe direcția determinată de $(0, 0)$ și (k_1, k_2).

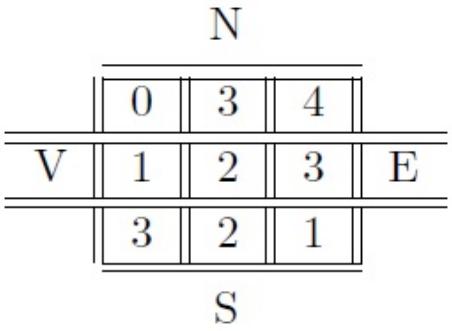


Figura 2.1

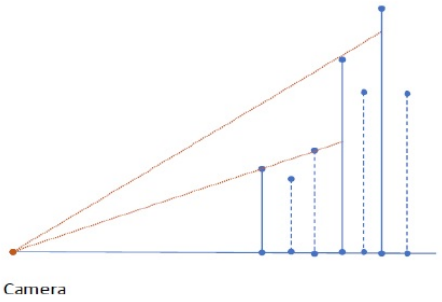


Figura 2.2

(c) (5p) Folosind funcția `viewSouthEast` și presupunând că este definită funcția `viewDir` (care are ca parametri matricea O, valorile k_1, k_2 și returnează numărul de clădiri vizibile pe direcția definită de (k_1, k_2)), descrieți funcția `viewAll` care returnează numărul total de clădiri care pot fi supravegheate atunci când camera este instalată în $(0, 0)$ la nivelul solului.

3. (15p) Fie P o permutare a numerelor de la 1 la n ($n \in \mathbb{N}$). Notăm cu $SUS(P)$ numărul minim de subșiruri crescătoare (elementele dintr-un subșir nu sunt neapărat consecutive în permutarea inițială) în care poate fi descompusă P. Notăm, de asemenea, cu $LDS(P)$ lungimea celui mai lung subșir descrescător al lui P.

   Exemplu: Fie $P = \{31254\}$. Atunci $SUS(P) = 2$, pentru că P nu e un șir crescător (prin urmare, $SUS(P) > 1$), dar putem descompune P în $P_1 = (35)$ și $P_2 = (124)$. De aemenea, $LDS(P) = 2$, pentru că P are subșirul descrescător (32) (de lungime 2), dar P nu are subșiruri descrescătoare de lungime 3.

   (a) (3p) Să se argumenteze că pentru orice permutare P, are loc $SUS(P) \geq LDS(P)$.

   (b) (10p) Scrieți un subprogram care construiește o descompunere a unei permutări P într-un număr minim de subșiruri crescătoare.

   (c) (2) Demonstrați, folosind rezultatul de la punctul (a), că subprogramul descris la punctul (b) realizează descompunerea lui P într-un număr minim de subșiruri crescătoare.

Notă:

1. Limbajul de programare este la alegere între C/C++ și Pascal.

2. Pentru fiecare dintre subprogramele (funcții, proceduri) scrise se vor specifica toate declarațiile de variabile globale și locale necesare și se vor pune comentarii explicative. **NU** este necesară scrierea programului principal și nici preluarea datelor prin citire.

BAREM

Start ..... **10 pct.**

Problema 1 ..... **50 pct.**

1a ..... 10 pct.

1(a)i. ..... 5 pct.

(declarații variabile locale/globale, definire corectă subprogram (inclusiv lista de parametri) - 1p; algoritm corect de extragere a cifrelor - 2p; algoritm corect de construire a tabloului c care indică corect cifrele comune - 2p)

1(a)ii. ..... 5 pct.

(declarații variabile locale/globale, definire corectă subprogram (inclusiv lista de parametri) - 1p; algoritm corect de extragere a cifrelor - 2p; algoritm corect de parcurgere a elementelor tabloului c - 2p; algoritm corect de construire a numărului z - 2p)

1b ..... 10 pct.

1(b)i. ..... 5 pct.

(lungimea laturii este $d = cmmd(m, n)$; numărul minim de tăieturi este $m/d-1$ (pe verticală), respectiv $n/d - 1$ (pe orizontală))

1(b)ii. ..... 5 pct.

(declarații variabile locale/globale, definire corectă subprogram (inclusiv lista de parametri) - 1p; algoritm corect de calcul a celui mai mare divizor comun - 3p; algoritm corect de calcul a numărului de tăieturi și afișarea rezultatelor - 1p)

1c ..... 5 pct.

1(c)i. ..... 2 pct.

A. 12; B. 12; C. $a * b$

1(c)ii. ..... 3 pct.

(definire corectă a funcției (inclusiv lista de parametri și returnare rezultat) - 1p; implementare corectă a funcției recursive - 2p)

1d ..... 5 pct.

(declarații variabile locale/globale, definire corectă subprogram (inclusiv lista de parametri) - 1p; parcurgere corectă a matricii și determinarea numărului maxim de prieteni (suma maximă pe linie sau coloană) - 2p; determinarea corectă și afișarea indicilor de linie (coloană) pentru care suma este maximă - 2p)

1e ..... 5 pct.

1(e)i. ..... 2 pct.

(Răspuns corect: C.)

1(e)ii. ..... 3 pct.

(Dacă $m = 1$ atunci este o singură posibilitate (o piesă de forma 2×1). Dacă ultima piesă completată este de forma 2×1 atunci sunt $V(m - 1)$ variante de

a completa primele $m-1$ coloane. Dacă ultima piesă completată este de forma 1×2 atunci penultima trebuie să fie de aceeași formă și sunt $V(m-2)$ variante de a completa primele $m-2$ coloane. Deci dacă $m > 2$ are loc $V(m) = V(m-1) + V(m-2)$.)

1f ..... 15 pct.

1(f)i. ..... 7 pct.

(declarații variabile locale/globale, definire corectă subprogram (inclusiv lista de parametri) - 1p; implementarea corectă a unui algoritm de sortare simultană a tablourilor x și y (cu proprietatea că $y[i]/x[i] \leq y[i+1]/x[i+1]$ - 6p)

1(f)ii. ..... 8 pct.

(declarații variabile locale/globale, definire corectă subprogram (inclusiv lista de parametri și returnarea rezultatului) - 1p; parcurgerea tablourilor sortate și contorizarea corectă a numărului de linii (de exemplu, contorizând numărul de elemente $(x[i], y[i])$ cu proprietatea că $y[i] * x[i+1] \neq y[i+1] * x[i]$ și adăugând 1) - 7p)

Problema 2 ..... **25 pct.**

2a ..... 15 pct.

(declarații variabile locale/globale, definire corectă subprogram (inclusiv lista de parametri) - 1p; algoritm corect de contorizare a clădirilor - 8p; implementarea corectă a parcurgerii clădirilor pe direcțiile SUD, EST, SUD-EST - 6p)

Idee de contorizare a numărului de clădiri vizibile pe o direcție dată (folosind proprietăți referitoare la asemănarea triungiurilor):

Pas 1. se pornește de la prima clădire pe direcția dată: $m \leftarrow 1$;

Pas 2. se parcurg clădirile k, până la prima clădire pentru care $h_k/d_k > h_m/d_m$ (sau până când s-au parcurs toate clădirile), unde h_i = înălțime clădire, iar d_i este distanța de la clădirea i la punctul unde se află camera (întrucât clădirile sunt la distanțe egale este suficient să se considere $d_i = i$; fiecare clădire k pentru care $h_k/d_k < h_m/d_m$ se contorizează ca fiind vizibilă.

Pas 3. dacă există o clădire k cu proprietatea $h_k/d_k > h_m/d_m$ atunci $m \leftarrow k$ (această clădire devine referință în raportul de asemănare) și se reia de la *Pas 2.*

2b ..... 5 pct.

(Regula de calcul a poziției clădirii i pe direcția definită de $(0,0)$ și (k_1, k_2) este: $(k_1 * i, k_2 * i)$ - 5p)

2c ..... 5 pct.

(declarații variabile locale/globale, definire corectă subprogram (inclusiv lista de parametri) - 1p; algoritm corect de contorizare și apeluri corecte ale funcțiilor `viewSouthEast, viewDir` - 4p)

Problema 3 ..... **15 pct.**

3a ..... 3 pct.

(Idee: Fie Q un subșir descrescător al lui Pde lungime $LDS(P)$. În orice descompunere a lui P în subșiruri crescătoare, termenii din Q trebuie să apară în subșiruri diferite. Prin urmare, $SUS(P) \geq LDS(P)$.)

3b ..... 10 pct.

(declarații variabile locale/globale, definire corectă subprogram (inclusiv lista de parametri) - 1p; parcurgere corectă a elementelor permutării - 1p; algoritm corect de construire a subșirurilor - 6p; completare corectă a subșirurilor în structura de date pentru care s-a optat (matrice, liste etc.) - 2p)

Idee: se poate aplica o strategie de tip "greedy": se parcurg elementele permutării și se adaugă fiecare la primul subșir compatibil (în care ultimul element adăugat este mai mic decât elementul curent), inițiind un nou subșir dacă adăugarea la unul existent nu este posibilă.

3c ..... 2 pct.

(Fie m numărul de subșiruri construite folosind algoritmul de la punctul (b). Vom demonstra că $m \leq LDS(P)$ (corectitudinea rezultând din proprietatea de la punctul (a) și această observație). Pentru aceasta e suficient să creăm un subșir descrescător care conține un element din fiecare subșir creat folosind algoritmul "greedy" de la punctul (b). Să considerăm ultimul element x_m adăugat în ultimul subșir S_m. x_m a fost adăugat în S_m (și nu în S_{m-1} pentru că a existat un element $x_{m-1} > x_m$ în S_{m-1} la momentul în care l-am adăugat pe x_m (x_{m-1} "nu a permis" adăugarea lui x_m în subșirul S_{m-1}). Pe de altă parte, x_{m-1} este în S_{m-1} pentru că există x_{m-2} în S_{m-2}, cu $x_{m-2} > x_{m-1}$ care "nu a permis" ca x_{m-1} să fie adăugat în subșirul S_{m-2} ș.a.m.d. În acest fel, am construit un șir descrescător $x_1 > x_2 > \cdots > x_m, x_i \in S_i$.)

2.4 Ediția a VI-a (2021)

1. Se consideră funcția T descrisă mai jos care primește un parametru întreg și returnează o valoare întreagă (operatorii DIV și MOD specifică câtul, respectiv restul împărțirii întregi).

```
funcție T(n)
    dacă n=1 atunci returnează 1 sf_dacă
    S ← 0
    m ← n DIV 2
    dacă n MOD 2 = 0 atunci
        m ← m − 1
    sf_dacă
    i ← 1
    cât_timp i <= m execută
        S ← S + T(i) * T(n − i)
        i ← i + 1
    sf_cât_timp
    dacă n MOD 2 = 0 atunci
        x ← T(n/2)
        returnează 2 * S + x * x
    altfel
        returnează 2 * S
    sf_dacă
sf_funcție
```

   Ce valoare va returna funcția dacă este apelată pentru $n = 5$?

   (a) 61
   (b) 23
   (c) 12
   (d) 14 Răspuns corect
   (e) 15

2. Se consideră următoarea secvență de prelucrări prin care se completează elementele unui tablou a:

```
pentru i ← 1, 200 execută
    a[i] ← i * i − i
sf_pentru
```

   Câte dintre elementele tabloului a au cifra unităților egală cu 0?

   (a) 20

(b) 21

(c) 79

(d) 80 Răspuns corect

(e) 81

(f) 82

Indicație: Se determină câte numere sunt multiplu de 5 de la 1 la 200 și de la 0 zero 199.

3. Pentru decorarea cu pitici artizanali a unei grădini avem la dispoziție 130 de lei. Știind că un pitic mare costă 50 lei, un pitic mijlociu costă 25 lei și un pitic mic costă 10 lei, câte variante de decorare pot fi identificate astfel încât să nu rămână mai mult de 9 lei neutilizați?

   (a) 12 Răspuns corect

   (b) 10

   (c) 13

   (d) 5

   Indicație: Se poate construi arborele de parcurgere a spațiului soluțiilor pornind de la nodul cu suma totală și ramificând prima dată după pitici mari (pot fi 2, 1 sau 0), dupa aceea pentru pitici medii (de restul banilor se cumpără pitici mici ca sa nu rămână rest mai mare ca 9):

   - 2 mari → 1 sau 0 medii (2 variante)
   - 1 mare → 3, 2, 1, 0 medii (4 variante)
   - 0 mari → 5, 4, 3, 2, 1, 0 medii (6 variante)

   În total 12 variante.

4. Câte zerouri are la sfârșit 2021! (factorialul numărului 2021)?

   (a) 202

   (b) 420

   (c) 500

   (d) 501

   (e) 503 Răspuns corect

   Indicație: Se vor număra numerele până la 2021 care sunt multipli de 5. Fiecare multiplu de 5 va adăuga un 0. Fiecare multiplu de 5^2 va adăuga câte un 0 în plus pentru că a fost deja numărat ca și multiplu de 5. Se va continua cu multiplii lui 5^3, apoi 5^4.

   2021 / 5 = 404

   2021 / 25 = 80

   2021 / 125 = 16

   2021 / 625 = 3

   total = 503

5. Se consideră secvența de instrucțiuni care transformă un tablou x ce conține n valori reale:

```
pentru i ← 1, n − 1 execută
    dacă x[i] > x[i + 1] atunci
        aux ← x[i]
        x[i] ← x[i + 1]
        x[i + 1] ← aux
    sf_dacă
sf_pentru
```

Care dintre următoarele proprietăți este satisfacută de către elementele tabloului x după execuția secvenței de mai sus?

(a) $x[n] \leq x[i]$ pentru orice $i \in \{1, 2, \ldots, n\}$
(b) $x[n] \geq x[i]$ pentru orice $i \in \{1, 2, \ldots, n\}$ Răspuns corect
(c) $x[i] \leq x[i + 1]$ pentru orice $i \in \{1, 2, \ldots, n - 1\}$
(d) $x[i] \geq x[i + 1]$ pentru orice $i \in \{1, 2, \ldots, n - 1\}$
(e) $x[1] \leq x[i]$ pentru orice $i \in \{1, 2, \ldots, n\}$
(f) $x[1] \geq x[i]$ pentru orice $i \in \{1, 2, \ldots, n\}$

Indicație: Prin interschimbarea elementelor vecine, valoarea maximă va fi plasată pe ultima poziție.

6. Se consideră un text stocat într-un șir de caractere, s, cu n elemente indexate începând cu 1. Se pune problema transformării șirului de caractere prin eliminarea spațiilor multiple (de exemplu șirul de caractere "un sir" este transformat în "un sir").

```
i ← 1
cât_timp i < n execută
    k ← 0
    cât_timp s[i] =' ' execută k ← k + 1; i ← i + 1; sf_cât_timp
    dacă k > 1 atunci
        ⟨ fragment pseudocod ⟩
        n ← n − k + 1
        i ← i − k
    sf_dacă
    i ← i + 1
sf_cât_timp
```

Care dintre fragmentele de cod asigură eliminarea spațiilor adiționale?

(a) Răspuns corect

$j \leftarrow i$
 `cât_timp` $j <= n$ `execută`
  $s[j-k+1] \leftarrow s[j]$
  $j \leftarrow j+1$
`sf_cât_timp`

(b) Răspuns corect

$j \leftarrow i$
$h \leftarrow i-k+1$
`cât_timp` $j <= n$ `execută`
  $s[h] \leftarrow s[j]$
  $j \leftarrow j+1$
  $h \leftarrow h+1$
`sf_cat_timp`

(c) $j \leftarrow i$
`cât_timp` $j <= n$ `execută`
  $s[j-k] \leftarrow s[j]$
  $j \leftarrow j+1$
`sf_cât_timp`

(d) Niciuna din variantele propuse nu este corectă

7. Se consideră o secvență de n valori naturale stocate într-un tablou x cu elemente indexate începând cu 1. În ipoteza că secvența conține exact o valoare egală cu 0 se dorește transformarea tabloului prin permutare circulară înspre stânga, astfel încât valoarea 0 să ajungă pe prima poziție (de exemplu secvența $(2,1,3,0,5,4)$ este transformată în $(0,5,4,2,1,3)$):

$i \leftarrow 1$
`cât_timp` $x[?a] \neq 0$ `execută`
  $aux \leftarrow ?b$
  `pentru` $i \leftarrow 2, n$ `execută`
    $x[?c] \leftarrow x[?d]$
  `sf_pentru`
  $x[?e] \leftarrow aux$
`sf_cât_timp`

Care dintre următoarele variante de completare a pseudocodului de mai sus asigură cerința problemei?

(a) $?a \to 1$, $?b \to x[1]$, $?c \to i-1$, $?d \to i$, $?e \to n$ Răspuns corect

(b) $?a \to i$, $?b \to x[1]$, $?c \to i-1$, $?d \to i$, $?e \to n$

(c) $?a \to 1$, $?b \to x[i]$, $?c \to i-1$, $?d \to i$, $?e \to n$

(d) $?a \to n$, $?b \to x[n]$, $?c \to i$, $?d \to i-1$, $?e \to 1$

(e) Niciuna din variante nu este corectă.

8. Pentru care din următoarele apeluri funcția F va returna valoarea 1?

VARIANTA C

```
int F(int n, int x)
      if (n<2) return 1;
      else if (n % 2 == 0)
           return F(n/x,x+1);
      else return 0;
```

VARIANTA PASCAL

```
function F(n,x:integer):integer;
      begin
      if (n<2) then F := 1
      else if (n MOD x=0) then
           F := F(n DIV x, x+1)
      else
           F := 0;
      end;
```

(a) $F(5040, 2)$ Răspuns corect

(b) $F(24, 2)$ Răspuns corect

(c) $F(25, 2)$

(d) $F(36, 2)$

9. Ce valoare va avea variabila k în urma execuției secvenței de prelucrări pentru un număr natural nenul n (operatorul DIV produce câtul împărțirii întregi, iar $[x]$ reprezintă partea întreaga a numărului x)?

$k \leftarrow 0$

`cât_timp` $n > 0$ `execută`

$n \leftarrow n/2$

$k \leftarrow k + 1$

`sf_cât_timp`

(a) $n/2$

(b) $[log_2 n]$

(c) n

(d) $[log_2 n] + 1$ Răspuns corect

10. Se consideră matricea

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Care este valoarea elementului de pe prima linie, prima coloană (elementul din stânga sus) din matricea $A^{100} = A * A * \ldots * A$ (de 100 de ori)

(a) 100

(b) 1

(c) elementul cu indicele 100 din șirul dat prin relația de recurentă: $f_0 = 1$, $f_1 = 1$, $f_n = f_{n-1} + f_{n-2}$ Răspuns corect

(d) elementul cu indicele 99 din șirul dat prin relația de recurentă: $f_0 = 1$, $f_1 = 1$, $f_n = f_{n-1} + f_{n-2}$

(e) 2^{100}

11. Se consideră funcția recursivă:

```
funcție F(a)
    dacă a <= 3
        returneaza 1
    altfel
        returneaza F(a − 1) + F(a − 2) + F(a − 3)
    sf_daca
sf_functie
```

De câte ori se calculează F(4) dacă primul apel este F(10)?

(a) 25

(b) 24 Răspuns corect

(c) 28

(d) 15

12. Ce va afișa subprogramul de mai jos în cazul apelului F(27)?

```
subprogram F(n)
    dacă n <= 0 atunci afișare(0)
    altfel
        afișare( n )
        F(n − 5)
        afișare( n )
    sf_dacă
sf_subprogram
```

(a) 0 2 7 12 17 22 27

(b) 27 22 17 12 7 2 0 2 7 12 17 22 27 Răspuns corect

(c) 27 22 17 12 7 2 -2 2 7 12 17 22 27

(d) 25 20 15 10 5 0 5 10 15 20 25

(e) 27 22 17 12 7 2 0

13. Se consideră un număr natural n constituit din $k+1$ cifre ($n = c_k c_{k-1} \ldots c_1 c_0$). Care sunt cifrele numărului m construit prin secvența de prelucrări de mai jos în ipoteza că k=3? (operatorii DIV și MOD permit calculul câtului respectiv restului împărțirii în domeniul numerelor întregi)

$p \leftarrow 1$

$m \leftarrow n \ \ MOD \ \ 10$

$n \leftarrow n \ \ DIV \ \ 10$

$q \leftarrow 1$

`cât_timp` $n \neq 0$ `execută`

    $q \leftarrow q + 1$

    `pentru` $i \leftarrow 1, q$ `execută`

        $p \leftarrow p * 10$

    `sf_pentru`

    $m \leftarrow (n \ \ MOD \ \ 10) * p + m$

    $n \leftarrow n \ \ DIV \ \ 10$

`sf_cât_timp`

(a) $c_0 0 c_1 0 c_2 0 c_3$

(b) $c_3 0 c_2 0 c_1 0 c_0$

(c) $c_3 00 c_2 0 c_1 c_0$

(d) $c_3 000 c_2 00 c_1 0 c_0$ Răspuns corect

(e) $c_0 000 c_1 00 c_2 0 c_3$

14. De câte ori se execută instrucțiunea $x \leftarrow x + 1$ în secvența de mai jos?

$x \leftarrow 0$

`pentru` $i \leftarrow 2, N$ `execută`

    `pentru` $j \leftarrow 1, i - 1$ `execută`

        $x \leftarrow x + 1$

    `sf_pentru`

`sf_pentru`

(a) $N(N-1)$

(b) $N(N-1)/2$ Răspuns corect

(c) $N(N+1)/2$

(d) $(N-1)(i-1)$

Indicație: Numărul de execuții ale liniei $x \leftarrow x+1$ este dat de calculul următoarei sume $\sum_{i=2}^{N}\sum_{j=1}^{i-1} 1 = \sum_{i=1}^{N-1} i$.

15. Se consideră un tabloul t cu n elemente indexate începând cu 1 și următoarea secvență de prelucrări aplicate asupra elementelor tabloului:

```
i ← 1
j ← n
cât_timp i < j execută
    cât_timp (t[i] > 0 si i <= n) execută i ← i + 1 sf_cât_timp
    cât_timp (t[j] < 0 si j >= 1) execută j ← j - 1 sf_cât_timp
    dacă i < j atunci
        aux ← t[i]
        t[i] ← t[j]
        t[j] ← aux
    sf_dacă
sf_cât_timp
```

Ce efect are secvența de prelucrări asupra tabloului t?

(a) Ordonează crescător tabloul t

(b) Plasează elementele pozitive în prima parte a tabloului t Răspuns corect

(c) Nu se poate spune nimic despre structura tabloului t

(d) Ordonează decrescător tabloul t

(e) Plasează elementele negative în prima parte a tabloului t

Problema 1

Se consideră un dreptunghi de dimensiune $2 \times n$ (două linii și n coloane) și se dorește acoperirea dreptunghiului cu piese de domino de forma 2×1 sau 1×2.

1. Propuneți o funcție care determină numărul de modalități de acoperire exactă a dreptunghiului cu piese (nu pot să rămână porțiuni neacoperite iar piesele nu pot depăși conturul dreptunghiului).
2. În ipoteza că două dintre celulele dreptunghiului aflate in colțuri opuse sunt inaccesibile (de exemplu celula $(1,1)$ și celula $(2,n)$ sau celule $(2,1)$ și celula $(1,n)$), propuneți o funcție care determină numărul de modalități de acoperire exactă a dreptunghiului cu piese (exceptând celulele din cele două colțuri).

Prezentați pe scurt ideea de rezolvare și descrieți soluția în pseudocod sau într-un limbaj de programare la alegere (C/C++, Pascal, Python).

Idee de rezolvare

1. Considerând dreptunghiul de dimensiune n se poate începe acoperirea fie cu:

   1 - o piesă plasată vertical, caz în care problema se reduce la o problemă de dimensiune $n-1$

   2 - două piese plasate orizontal, caz în care problema se reduce la o problemă de dimensiune $n-2$

   Prin urmare formula de calcul a numărului de variante posibile este:

   C(1) = 1

   C(2) = 2 (se pot pune orizontal sau vertical)

   C(n) = C(n-1) + C(n-2) (de tip Fibonacci)

2. Se analizează paritatea lui n, dacă n este par nu există varianta de acoperire fără să se suprapună piesele. Dacă n este impar există o singura posibilitate de acoperire cu piese plasate orizontal (nu se poate pune nicio piesă verticală).

BAREM

Start ..... 1 pct.

(a) Descrierea ideii de rezolvare ..... 3 pct.

Implementarea funcției:

Declarații de variabile ..... 1 pct.

Implementare corectă $C(1) = 1$, $C(2) = 2$, $C(n) = C(n-1) + C(n-2)$ 2 pct.

Definirea corectă a funcției și returnarea rezultatului ..... 1 pct.

(b) Descrierea ideii de rezolvare ..... 1 pct.

Declarații, definire și implementare corectă a subprogramului ..... 1 pct.

Problema 2

Se consideră două secvențe de câte n valori naturale a_1, a_2, ..., a_n respectiv b_1, b_2, ..., b_n. Se pune problema determinării unei permutări de ordin n, $p = (p(1), p(2), \ldots, p(n))$ astfel încât suma $|a_1 - b_{p(1)}| + |a_2 - b_{p(2)}| + \ldots + |a_n - b_{p(n)}|$ să fie cât mai mare. Descrieți un algoritm care permite obținerea permutării și sumei corespunzătoare.

De exemplu, pentru $a = (5, 9, 12, 4)$ și $b = (7, 2, 15, 5)$ o permutare care satisface cerința este $p = (1, 4, 2, 3)$, suma fiind $|5 - 7| + |9 - 5| + |12 - 2| + |4 - 15| = 27$.

Prezentați pe scurt ideea de rezolvare, justificați că ideea este corectă și descrieți soluția în pseudocod sau într-un limbaj de programare la alegere (C/C++, Pascal, Python).

Idei de rezolvare

Varianta 1: Pentru construirea permutării se parcurg elementele lui a în ordine descrescătoare și a lui b în ordine crescătoare. La primul pas se va completa în permutare $p[poz_{max_a}] = poz_{min_b}$ și se continua procesul.

Varianta 2: Generarea tuturor permutărilor și selectarea celei de sumă maximă (varianta aceasta este ineficientă).

BAREM

Start ..... 1 pct.

Idee rezolvare - specificare modului de obține al permutării ..... 2 pct.

Argumentarea corectitudinii algoritmului propus ..... 3 pct.

Declarații de variabile care descriu structurile utilizate ..... 1 pct.

Construirea corectă a permutarii ..... 2 pct.

Calculul corect al sumei ..... 1 pct.

Problema 3

Pentru un grup de n persoane se cunosc perechile de persoane care intră frecvent în contact (fac parte din aceeași familie sau lucrează împreună). În contextul răspândirii unei epidemii se consideră că toate persoanele care au intrat în contact cu o persoană infectată devin infectate. Cunoscând care este pacientul 0 determinați numărul de persoane care au fost infectate.

Exemplu: $n = 7$ și perechile de persoane care intră în contact sunt: (P_1, P_2), (P_1, P_4), (P_1, P_7), (P_3, P_6) și (P_5, P_6). În ipoteza că pacientul zero este P_2 modul de răspândire al infecție este: la etapa 1 se infectează P_1, la etapa 2 se infectează P_4 și P_7 și procesul de răspândire a infecției se oprește deoarece nu mai există persoane neinfectate care au intrat în contact cu o persoană infectată, numărul de persoane infectate fiind 4.

Prezentați pe scurt ideea de rezolvare și descrieți soluția în pseudocod sau într-un limbaj de programare la alegere (C/C++, Pascal, Python).

Idei de rezolvare

Varianta 1: Relațiile dintre persoane se pot reprezenta sub forma unui graf, iar numărul de noduri (determinat prin parcurgere în adancime sau în lățime) din subgraful care conține persoana zero reprezintă soluția problemei.

Varianta 2: Pentru a reprezenta contactele se poate folosi o matrice (m), $m_{i,j} = 1$ dacă persoana i a intrat în contact cu persoana j. Pentru a determina numărul de persoane infectate se poate folosi un vector (v) care stochează informația dacă persoana este infectată sau nu. Se pornește de la persoana zero (p_0) și se caută persoanele cu care este în contact $m_{p_0,k} = 1$, $k = 1, n$. În prima etapă se marchează ca infectate toate persoanele aflate în contact direct cu pacientul p_0. La etapa următoare se marchează ca infectate toate persoanele neinfectate care sunt în contact direct cu cele infectate la etapa anterioară. Procesul va continua până când nu mai apar persoane nou infectate.

BAREM

Start ..... 1 pct.

Idee rezolvare - descrierea procesului de răspândire a infecției ..... 2 pct.

Descrierea structurilor de date folosite și declarații ..... 2 pct.

Implementarea algoritmului de răspândire a infecției ..... 4 pct.

Determinarea corectă a numărului de persoane infectate ..... 1 pct.

Capitolul 3

Subiecte date la examenul de admitere la facultate

3.1 Sesiunea Iulie 2016

1. (a) (5p) Scrieți o funcție C/C++/Pascal care primește ca parametru o valoare naturală n din mulțimea $\{0, \ldots, 255\}$ și returnează valoarea m calculată după regula:

$$m = \begin{cases} \text{n/2} & \text{dacă } n \text{ este par} \\ \text{128+(n-1)/2} & \text{dacă } n \text{ este impar} \end{cases}$$

   (b) (5p) Se consideră un pătrat de latură L al cărui vârf din stânga jos are coordonatele (x_0, y_0). Se pune problema verificării dacă un punct de coordonate (x, y) se află sau nu în interiorul pătratului. Scrieți o funcție C/C++/Pascal care primește ca parametri de intrare valorile reale L, x_0, y_0, x, y și returnează 1 dacă punctul se află în interiorul pătratului și 0 în caz contrar.

   (c) (5p) Scrieți o funcție C/C++/Pascal care primește 6 parametri (a, b, c, x, y, z) care reprezintă lungimile laturilor a două triunghiuri (a, b, c reprezintă lungimile laturilor primului triunghi, iar x, y, z lungimile laturilor celui de al doilea triunghi) și returnează 1 dacă triunghiurile sunt asemenea și 0 în caz contrar. Lungimile laturilor fiecărui triunghi sunt numere naturale și sunt enumerate într-o ordine arbitrară.

   (d) (5p) Se consideră o tablă caroiată ce conține $N \times N$ ($N > 3$) pătrățele din care se elimină două pătrățele aflate în extremitățile uneia dintre diagonale. Se dorește acoperirea tablei cu piese de domino de dimensiune 2×1 (care pot fi plasate pe orizontală sau verticală).

   Când este posibilă acoperirea exactă a tablei cu piese?

   i. când N este impar;
   ii. când N este par;
   iii. pentru nicio valoare a lui N;
   iv. pentru orice valoare a lui N.

2. Adresele de e-mail de la UVT au următoarea structură: `prenume.nume@e-uvt.ro` (numele și prenumele conțin doar litere mici).

   (a) (10p) Scrieți o funcție C/C++/Pascal care primește ca parametru un șir de caractere ce conține simbolul @ și returnează 1 dacă șirul reprezintă o adresă de e-mail care respectă structura de la UVT și 0 în caz contrar.

   (b) (10p) Scrieți o funcție (procedură) C/C++/Pascal care primește ca parametru un șir de caractere ce reprezintă o adresă de e-mail de la UVT corectă și afișează prenumele și numele separate de un spațiu, fiecare dintre ele începând cu majusculă.

   Exemplu: pentru adresa `ioan.popescu@e-uvt.ro` se va afișa `Ioan Popescu`.

3. Doi prieteni, Geo și Leo, joacă următorul joc: pe o foaie este scrisă o listă de M numere naturale nenule mai mici decât 10000. La începutul jocului, Geo și Leo aleg fiecare câte trei numere prime impare distincte, mai mici decât 100. La fiecare rundă se trece la următorul număr din listă, notat cu N, și fiecare dintre cei doi jucători își recalculează scorul după următoarea regulă: se adaugă câte 1 punct pentru fiecare factor din descompunerea în produs de factori primi a numărului N care se regăsește printre cele 3 numere prime pe care le-a ales la început, se scade câte un punct pentru fiecare dintre numerele sale prime care nu apare în descompunerea în produs de factori primi a numărului N. Câștigă jucătorul care are punctajul cel mai mic la terminarea tuturor numerelor de pe foaie.

   Exemplu: dacă numărul N este 150, iar Geo a ales la început numerele prime $3, 5$ și 13, va primi un total de 2 puncte: 1 punct pentru 3, 2 puncte pentru 5 și -1 punct pentru 13.

   Scrieți următoarele funcții (C/C++/Pascal) utile pentru calculul scorului ununi jucător:

   (a) (5p) `estePrim(x)` - o funcție care returnează 1 dacă `x` este prim și impar, respectiv 0 în caz contrar.

   (b) (5p) `esteFactorPrim(N, x)` - o funcție care returnează 1 dacă `x` apare în descompunerea în produs de factori primi a lui `N`, respectiv 0 în caz contrar; se va utiliza funcția `estePrim` descrisă la punctul (a).

   (c) (5p) `puncte(N, x)` - o funcție care returnează numărul de puncte primite pentru perechea (`N`, `x`); se va utiliza funcția `esteFactorPrim` descrisă la punctul (b).

   (d) (5p) `punctaj(listaNumere, nr1, nr2, nr3)` - o funcție care returnează scorul corespunzător listei de numere pentru jucătorul care a ales numerele prime inițiale `nr1, nr2` și `nr3`, determinat folosind funcții dintre cele descrise anterior.

4. Se consideră o imagine pe niveluri de gri, A, reprezentată printr-o matrice cu 50 de linii și 50 de coloana care conține valori din mulțimea $\{0, \ldots, 255\}$ (valoarea unui element din matrice reprezintă nivelul de gri al pixelului corespunzător din imagine).

(a) (10p) Scrieți o funcție (procedură) C/C++/Pascal care, pornind de la matricea A (variabilă globală), construiește histograma imaginii (un tablou unidimensional H cu 256 de elemente în care elementul de pe poziția i conține numărul de elemente din matricea A care au valoarea i).

(b) (10p) Scrieți o funcție C/C++/Pascal care primește ca parametru tabloul cu histograma imaginii determinată la punctul (a) și returnează nivelul mediu de gri, g, (calculat ca medie ponderată a elementelor din matricea A: $g = (H[0]\times 0 + H[1]\times 1 + \cdots + H[255]\times 255)/(H[0]+H[1]+\cdots+H[255])$).

(c) (10p) Scrieți o funcție (procedură) C/C++/Pascal care primește ca parametri matricea A și nivelul mediu de gri g, determinat la punctul (b), și construiește o matrice B de aceleași dimensiuni cu A cu proprietatea că elementul de pe linia i și coloana j este egal cu 1 dacă elementul corespunzător din matricea A este mai mare sau egal cu g, respectiv este egal cu 0 dacă elementul corespunzător din A este mai mic decât g.

BAREM

Start ..... **10 pct.**

Problema 1 ..... **20 pct.**

1a ..... 5 pct.

(declararea corectă a funcției și returnarea rezultatului - 1p; calculul corect al valorii m - 4p)

1b ..... 5 pct.

(declararea corectă a funcției și returnarea rezultatului - 1p; specificarea corectă a condiției de punct interior ($x \in (x_0, x_0 + L)$ și $y \in (y_0, y_0 + L)$) - 4p)

1c ..... 5 pct.

(declararea corectă a funcției și returnarea rezultatului - 1p; verificarea corectă a condiției de asemănare a triunghiurilor (se transformă variabilele astfel încât $a \leq b \leq c$ și $x \leq y \leq z$ după care se verifică condiția de proporționalitate a lungimilor laturilor: $a/x = b/y = c/z$ sau $a \times y = b \times x$ și $b \times z = c \times y$) - 4p)

1d (iii) ..... 5 pct.

Problema 2 ..... **20 pct.**

2a ..... 10 pct.

(declararea corectă a funcției și returnarea rezultatului - 1p; verificarea corectă a structurii de adresă UVT (subșirurile ce corespund prenumelui și numelui sunt nevide și conțin doar litere mici - 2p, există un singur punct înainte de simbolul @ - 2p, iar subșirul de după simbolul @ coincide cu "e-uvt.ro" - 5p))

2b ..... 10 pct.

(declararea corectă a funcției (procedurii)- 1p; afișarea prenumelui - 4p și numelui - 4p începând cu majuscule - 1p)

Problema 3 ..... **20 pct.**

3a ..... 5 pct.

(declararea corectă a funcției și returnarea rezultatului - 1p; verificarea corectă a proprietății de număr impar - 1p și de număr prim - 3p)

3b ..... 5 pct.

(declararea corectă a funcției și returnarea rezultatului - 1p; verificarea corectă a faptului că x este factor prim al lui N - 4p)

3c ..... 5 pct.

(declararea corectă a funcției și returnarea rezultatului - 1p; calculul corect al numărului de puncte - 4p)

3d ..... 5 pct.

(declararea corectă a funcției și returnarea rezultatului - 1p; calculul corect al scorului: parcurgerea listei cu numere - 2p, apelul corect al funcției punctaj - 2p)

Problema 4 ..... **30 pct.**

4a ..... 10 pct.

(declararea corectă a matricii A - 1p; declararea corectă a tabloului H - 1p; declararea corectă a funcției (procedurii) - 1p; completarea corectă a tabloului H - 7p)

4b ..... 10 pct.

(declararea corectă a funcției și returnarea rezultatului - 1p; calculul corect al mediei ponderate - 9p)

4c ..... 10 pct.

(declararea corectă a matricii B - 1p; declararea corectă a funcției (procedurii) - 1p; construirea corectă a matricii B - 8p)

3.2 Sesiunea Iulie 2017

1. (10p) O matrice pătratică se consideră matrice de tip X dacă toate elementele sale sunt nule cu excepția celor care se află pe una dintre cele două diagonale.

   (a) (5p) Pentru o matrice pătratică A de dimensiune n ($n \leq 10$) se consideră următorul algoritm al cărui scop este să verifice dacă A este matrice de tip X :

   ```
   corect ← 1
   pentru i ← 1, n execută
      pentru j ← 1, n execută
        dacă < ... condiție referitoare la A, i și j ··· >
                                                  atunci corect ← 0
   ```

   Completați condiția (< ... condiție referitoare la A, i și $j \cdots >$) din pseudocod astfel încât variabila `corect` să aibă valoarea 1 dacă matricea este de tip X și valoarea 0 altfel.

   (b) (5p) Scrieți o funcție C/C++/Pascal care implementează algoritmul de la punctul (a) și returnează valoarea variabilei corect.

2. (10p) Se consideră următoarea relație de recurență (pentru n și k numere naturale):

$$f(n,k) = \begin{cases} 0 & \text{dacă } n < k \text{ sau } n = 0 \\ 1 & \text{dacă } n > 0 \text{ și } (n = k \text{ sau } k = 0) \\ \text{f(n-1,k)+f(n-1,k-1)} & \text{dacă } n > 0 \text{ și } k < n \end{cases}$$

   (a) (2p) Ce valoare are $f(4, 2)$?

   (b) (3p) Ce calcul descrie relația de recurență (pentru $1 \leq k \leq n$)? Alegeți raspunsul corect dintre:

   i. $\sum_{i=k}^{n-1} i + \sum_{i=k-1}^{n-1} i$;
   ii. C_n^k;
   iii. $2\sum_{i=1}^{n-1} i + \sum_{i=1}^{k-1} i + \sum_{i=1}^{k} i$;
   iv. A_n^k

   (c) (5p) Scrieți o funcție recursivă în C/C++/Pascal care implementează relația de recurență (primește parametrii n și k și returnează valoarea lui $f(n, k)$).

3. (35p) Se consideră o secvență cu k ($k \leq 1000$) numere naturale stocată într-un tablou unidimensional S.

   (a) (10p) Scrieți o funcție C/C++/Pascal care primește ca parametri pe k și S și efectuează următoarele prelucrări:

   i. completează o variabilă globală L (tablou unidimensional) care conține numărul de elemente din fiecare subsecvență cu valori de aceeași paritate;

ii. returnează numărul N de subsecvențe identificate
Exemplu: Pentru secvența $(5, 1, 3, 2, 4, 7, 6, 4, 10, 8, 3)$ tabloul L va conține valorile $(3, 2, 1, 4, 1)$, iar funcția va returna 5.

(b) (10p) Scrieți un subprogram C/C++/Pascal care primește ca parametri indicii, i_1 și i_2 (se presupune că $i_1 \leq i_2$, a două elemente din S și ordonează crescător subtabloul din tabloul S (considerat variabilă globală) determinat de cei doi indici.
Exemplu: În cazul în care indicii pornesc de la 0, pentru secvența $(5, 1, 3, 2, 4, 7, 6, 4, 10, 8, 3)$ și $i_1 = 6, i_2 = 9$, după ordonare, variabila S va conține $(5, 1, 3, 2, 4, 7, 4, 6, 8, 10, 3)$.

(c) (10p) Folosind rezultatele obținute apelând funcția de la punctul (a) și apelând subprogramul de la punctul (b) să se transforme tabloul S astfel încât toate subsecvențele de numere de aceeași paritate să fie ordonate crescător. Secvența $(5, 1, 3, 2, 4, 7, 6, 4, 10, 8, 3)$ se va transforma în $(1, 3, 5, 2, 4, 7, 4, 6, 8, 10, 3)$.

(d) (5p) Ana și Maria joacă următorul joc: fiind dată o secvență de numere naturale fiecare jucător trebuie să elimine subsecveța de numere de aceeași paritate de la începutul secvenței curente. Pentru secvența $(5, 1, 3, 2, 4, 7, 6, 4, 10, 8, 3)$, după mutarea primului jucător, secvența devine $(2, 4, 7, 6, 4, 10, 8, 3)$, după mutarea celui de al doilea, devine $(7, 6, 4, 10, 8, 3)$ etc. Jucătorul care elimină ultima subsecvență pierde jocul. Ambele jucătoare au acces la întreaga secvență. Ana are dreptul să decidă dacă vrea să înceapă jocul sau dacă o lasă pe Maria să înceapă. Ce informații despre secvența sunt suficiente pentru Ana pentru a putea lua decizia care garantează că va câștiga? Descrieți, în limbaj natural, regula de decizie corespunzătoare.

4. (35p) Un careu Sudoku este un tablou bidimensional cu 9 linii și 9 coloane ce conține cifre cuprinse între 1 și 9. Careul este considerat corect completat dacă satisface următoarele proprietăți (vezi exemplul de mai jos):

(i) fiecare linie și fiecare coloană conține cifre distincte;

(ii) fiecare dintre cele 9 subtablouri cu 3 linii și 3 coloane (care acoperă în mod disjunct tabloul de 9×9) conține cifre disctincte.

(a) (10p) Scrieți o funcție C/C++/Pascal care primește ca parametru un tablou unidimensional cu 9 elemente și verifică dacă elementele sunt cifre distincte (se presupune că elementele sunt numere naturale între 1 și 9). Funcția va returna valoarea 1 dacă tabloul satisface proprietatea cerută și 0 în caz contrar.

(b) (2p) Declarați o variabilă cu numele S în care să poată fi stocat tabloul bidimensional cu 9 linii și 9 coloane.

(c) (5p) Scrieți o funcție C/C++/Pascal care primește ca parametru indicii corespunzători celulei din stânga sus, respectiv ale celulei din dreapta jos a unui subtablou de 3×3 elemente din tabloul S și verifică dacă elementele subtabloului specificat sunt cifre distincte (se presupune că elementele sunt numere naturale între 1 și 9). Funcția va returna valoarea 1 dacă subtabloul specificat satisface proprietatea cerută și 0 în caz contrar. Este permisă utilizarea funcției de la punctul (a).

(d) (15p) Scrieți un subprogram C/C++/Pascal care folosește funcțiile definite la (a) și (c) și care verifică dacă tabloul stocat în variabila globală S corespunde unui careu Sudoku completat corect. Subprogramul va afișa "Careu corect" dacă sunt satisfăcute proprietățile, respectiv "Careu incorect" dacă nu sunt satisfăcute.

(e) (3p) Presupunând că într-un subtablou 3×3 sunt completate k elemente ($1 < k < 9$) cu valori specificate, care este numărul de subtablouri distincte (care satisfac condiția de Sudoku) ce pot fi obținute prin completarea celor $9 - k$ elemente? Alegeți răspunsul corect dintre:

i. A_9^k;
ii. A_9^{9-k};
iii. C_9^k;
iv. C_9^{9-k};
v. $k!$;
vi. $(9-k)!$.

Exemplu de careu completat corect:

6	8	2	1	9	4	3	5	7
7	3	1	5	6	8	9	2	4
4	9	5	7	2	3	8	6	1
8	2	7	9	3	5	1	4	6
5	1	9	6	4	7	2	8	3
3	6	4	2	8	1	5	7	9
9	5	6	4	1	2	7	3	8
2	4	8	3	7	9	6	1	5
1	7	3	8	5	6	4	9	2

Sursa:

`http://www.rasfoiesc.com/familie/copii/Ghid-Sudoku-Regulile-jocului69.php`

BAREM

Start ..... **10 pct.**

Problema 1 ..... **10 pct.**

1a ..... 5 pct.

(Condiția $((i = j$ sau $i + j = n + 1)$ și $A_{ij} = 0)$ sau $(i \neq j$ și $i + j \neq n + 1$ și $A_{ij} \neq 0))$

1b ..... 5 pct.

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; descrierea corectă a parcurgerii matricii - 2p; specificarea corectă a condiției și completarea variabilei corect - 1p)

Problema 2 ..... **10 pct.**

2a ..... 2 pct.

Răspuns corect: 6

2b ..... 3 pct.

Răspuns corect: *ii*.

2c ..... 5 pct.

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p;implementarea corectă a relației de recurență - 4p)

Problema 3 ..... **35 pct.**

3a ..... 10 pct.

(declararea corectă a funcției și a variabilelor locale - 2p; completarea corectă a tabloului L - 6p; determinarea și returnarea numărului de subsecvențe - 2p)

3b ..... 10 pct.

(declararea corectă a subprogramului și a variabilelor locale - 1p; implementarea corectă a algoritmului de sortare - 9p)

3c ..... 10 pct.

(declararea variabilelor utilizate - 1p; parcurgerea tabloului S - 3p; stabilirea indicilor corespunzători și apelul corect a subprogramului de la punctul (b) - 6p)

3d ..... 5 pct.

Răspuns: este suficient să se cunoască numărul de subsecvențe de valori cu aceeași paritate (valoarea N); regula de decizie: "dacă N este par atunci Ana începe jocul, altfel o lasă pe Maria să înceapă".

Problema 4 ..... **35 pct.**

4a ..... 10 pct.

Variantă de rezolvare: se construiește un tabel de frecvențe corespunzător naturale cuprinse între 1 și 9 și se verifică dacă toate elementele tabloului de frecvențe sunt egale cu 1.

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; verificarea condiției că elementele au valori distincte - 9p)

4b ..... 2 pct.

(declararea corectă a tabloului bidimensional S - 2p)

4c ..... 5 pct.

Variantă de rezolvare: se construiește un tablou unidimensional conținând elementele subtabloului și se apelează funcția de la punctul (a)

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; parcurgerea corectă a elementelor subtabloului - 2p; verificarea condiției că elementele au valori distincte - 2p)

4d ..... 15 pct.

(declararea corectă a subprogramului și a variabilelor locale - 1p; verificarea proprietății la nivel de linii - 3p; verificarea proprietății la nivel de coloane - 3p; verificarea proprietății la nivel de subtablouri - 8p)

4e ..... 3 pct.

Răspuns: *vi.* $(9 - k)$!

3.3 Sesiunea Iulie 2018

1. (40p)

   (a) (5p) Se consideră suma $S = 1 - 1/3 + 1/5 - 1/7 + 1/9 - \ldots$ (primul termen este 1, al doilea termen este $-1/3$ etc.).

      i. Care este expresia corespunzătoare celui de al $k-$lea termen?

         A. $1/(2k-1)$;

         B. $1/(2k+1)$;

         C. $(-1)^k/(2k-1)$;

         D. $(-1)^{k+1}/(2k-1)$;

         E. $(-1)^{k+1}/(2k+1)$.

      ii. Scrieți o funcție în C/C++/Pascal care primește ca parametru un număr natural $n \geq 2$ și returnează suma primilor n termeni din S.

   (b) (5p) Un număr natural este considerat tip "dublu-unu" dacă reprezentarea sa în baza 2 conține exact o pereche de cifre consecutive egale cu 1 (numerele $3 = (11)_2, 6 = (110)_2$ sunt de tip "dublu-unu", dar numerele $5 = (101)_2$ și $7 = (111)_2$ nu sunt de acest tip).

      i. Câte numere de tip "dublu-unu" sunt în intervalul $(0, 2^d)$?

      ii. Care dintre următoarele afirmații este adevărată pentru orice număr n de tip "dublu-unu" din $(0, 2^d)$:

         A. există $1 \leq k \leq d$ astfel încât $n = 2^k - 2^{k-1}$;

         B. există $1 \leq k < d$ astfel încât $n = 2^k + 2^{k-1}$;

         C. există $0 < k < d$ astfel încât $n = 2^{k+1} + 2^{k-1}$;

         D. există $1 \leq k < d$ astfel încât $n = 2^k + 1$.

   (c) (10p) Se consideră un serviciu web la care utilizatorii se conectează/deconectează și se pune problema determinării numărului maxim de utilizatori conectați simultan pornind de la o secvență de semnale de forma: 1 (s-a conectat un utilizator), 0 (s-a deconectat un utilizator). De exemplu, pentru secvența $1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0$ numărul maxim de utilizatori conectați simultan este 5. Scrieți o funcție C/C++/Pascal care primește un tablou de semnale precum și numărul acestora și returnează numărul maxim de utilizatori conectați simultan.

   (d) (10p) Pentru un număr natural nenul n se pune problema determinării numărului de zerouri finale (de exemplu, 304500 conține două zerouri finale, 30245 conține 0 zerouri finale).

      i. Scrieți o funcție C/C++/Pascal (*nerecursivă*) care primește ca parametru numărul n și returnează numărul de zerouri finale.

      ii. Scrieți o funcție C/C++/Pascal *recursivă* care efectuează aceeași prelucrare.

   (e) (10p) Se consideră că un text a fost tehnoredactat folosind o tastatură defectă care "multiplică" simbolurile tastate și se pune problema "curățirii" acestui text prin eliminarea caracterelor în exces (se presupune că textul conține doar cuvinte în care nu intervin simboluri consecutive identice).

De exemplu, textul "Acccceessttta eesttte uuun exxeeemmpplluu" trebuie transformat în "Acesta este un exemplu". Considerând că textul este stocat într-un tablou cu elemente de tip caracter scrieți:

i. o funcție/procedură care construiește un nou tablou de caractere care conține textul curățat (pentru exemplul de mai sus noul tablou va avea 22 de caractere și va conține "Acesta este un exemplu");

ii. o funcție/procedură care transformă tabloul inițial astfel încât pe primele poziții să fie textul curat (în exemplul de mai sus noul conținut al tabloului este "Acesta este un exempluuuun exxeeemmpplluu"). Nu este permisă utilizarea unui tablou adițional.

2. (25p) Se consideră un set de n orașe, numerotate de la 1 la n ($n \leq 50$), și o "hartă" a conexiunilor aeriene dintre aceste orașe stocată sub forma unei matrici A cu n linii și n coloane în care elementul a_{ij} este 1 dacă există conexiune directă între orașele i și j și este 0 dacă nu există conexiune directă (se presupune că $a_{ii} = 0$ și $a_{ij} = a_{ji}$ pentru orice i și j din mulțimea $\{1, 2, \ldots, n\}$).

Exemplu: Se consideră 5 orașe și matricea de conectivitate directă:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

(a) Scrieți o funcție/procedură care construiește tabloul D care conține pentru fiecare oraș numărul de conexiuni directe ($D[i]$ conține numărul de conexiuni directe care pornesc din orașul i). *Exemplu:* $D = [2, 3, 2, 3, 4]$.

(b) Scrieți o funcție/procedură care primește ca parametru tabloul D și afișează orașele în ordinea descrescătoare a numărului de conexiuni directe. *Exemplu:* valorile afișate sunt $5, 2, 4, 1, 3$.

(c) Scrieți o funcție/procedură care primește ca parametri matricea conexiunilor și indicii a două orașe (i și j) și afișează toate variantele de a ajunge din orașul i în orașul j făcând cel mult o escală. *Exemplu:* pentru $i = 2, j = 5$ se afișează: $(2, 5), (2, 1, 5), (2, 4, 5)$.

(d) Scrieți o funcție/procedură care calculează matricea $B = A^2 = A \times A$.

(e) În ipoteza că fiecare oraș este conectat direct cu cel puțin un alt oraș, care dintre următoarele afirmații, referitoare la elementele matricii B, este adevărată?

i. $b_{ii} = 0$ și $b_{ij} = b_{ji}$ pentru orice i și j din mulțimea $\{1, 2, \ldots, n\}$;

ii. $b_{ii} > 1$ pentru orice i din mulțimea $\{1, 2, \ldots, n\}$ și există cel puțin o pereche de (i, j) cu proprietatea că $b_{ij} \neq b_{ji}$;

iii. b_{ii} conține numărul de conexiuni directe ale orașului i. Argumentați răspunsul selectat.

3. (25p) Un lucrător preia pachete (de dimensiuni diferite) de pe o bandă și le stivuiește astfel încât să nu pună niciodată un pachet peste unul de dimensiune mai mică (dacă un pachet nu poate fi plasat pe niciuna dintre stivele existente

este creată o nouă stivă). De exemplu, dacă dimensiunile pachetelor, în ordinea în care sosesc pe bandă, sunt $[3, 1, 2, 6, 5, 7, 4]$ atunci stivele constituite sunt $[3, 1], [2], [6, 5, 4], [7]$.

(a) Propuneți o structură de date pentru reprezentarea stivelor și scrieți o funcție/procedură care primește ca parametri numărul de pachete și un tablou cu dimensiunile acestora și care distribuie pachetele pe stive astfel încât numărul de stive create să fie cât mai mic (fiecare pachet este plasat pe prima stivă pe care este posibil).

(b) Presupunând că numărul de stive create este k justificați faptul că cel mai lung subșir strict crescător din tabloul dimensiunilor are k elemente (elementele din subșir nu sunt neapărat consecutive, dat trebuie să fie în aceeași ordine ca în șirul inițial; de exemplu, $[1, 2, 6, 7]$ este un subșir strict crescător al șirului $[3, 1, 2, 6, 5, 7, 4]$). Scrieți o funcție/procedură care construiește un subșir strict crescător de lungime maximă.

BAREM

Start ..... **10 pct.**

Problema 1 ..... **40 pct.**

1a ..... 5 pct.

1(a)i. ..... 2 pct.

(Răspuns corect: D.)

1(a)ii. ..... 3 pct.

(declararea corectă a funcției, a variabilelor locale și returnarea rezultatului - 1p; descrierea corectă a calculului sumei - 2p)

1b ..... 5 pct.

1(b)i. ..... 3 pct.

(Răspuns corect: $d - 1$)

1(b)ii. ..... 3 pct.

(Răspuns corect: B.)

1c ..... 10 pct.

(declararea corectă a funcției, a variabilelor de lucru și returnarea rezultatului - 1p; parcurgerea corectă a tabloului cu semnale - 2p; actualizarea corectă a numărului de utilizatori activi - 3p; determinarea valorii maxime - 4p)

1d ..... 10 pct.

1(d)i. ..... 5 pct.

(declararea corectă a funcției, a variabilelor de lucru și returnarea rezultatului - 1p; calculul corect al numărului de zerouri finale - 4p)

1(d)ii. ..... 3 pct.

(declararea corectă a funcției, a variabilelor de lucru și returnarea rezultatului - 1p; condiție corectă pentru ieșirea din apelul recursiv - 1p; apel recursiv corect - 1p; calculul corect al numărului de zerouri finale - 2p)

1e ..... 10 pct.

1(e)i. ..... 5 pct.

(declararea funcției/procedurii și a variabilelor utilizate - 1p; construirea corectă a noului tablou - 4p)

1(e)ii. ..... 3 pct.

(declararea funcției/procedurii și a variabilelor utilizate - 1p; transformarea corectă a tabloului - 4p)

Problema 2 ..... **25 pct.**

2a ..... 5 pct.

(declararea corectă a funcției/procedurii și a variabilelor utilizate - 1p; calculul sumei elementelor de pe fiecare linie (sau determinarea numărului de valori egale cu 1 de pe fiecare linie) și stocarea corectă în tabloul D - 4p)

2b ..... 8 pct.

(declararea corectă a funcției/procedurii și a variabilelor utilizate - 1p; implementarea corectă a unei metode de sortare - 5p; asigurarea concordanței dintre elementele tabloului D și numerele de ordine ale orașelor - 1p; afișarea numerelor de ordine ale orașelor în ordinea cerută - 1p)

2c ..... 5 pct.

(declararea corectă a funcției/procedurii și a variabilelor utilizate 1p; afișarea traseului direct (dacă există) - 1p; afișarea tuturor traseelor cu o escală - 3p)

2d ..... 5 pct.

(declararea corectă a funcției/procedurii și a variabilelor utilizate 1p; calculul corect al produsului - 4p)

2e ..... 2 pct.

(răspuns corect: iii. - 1p; argumentare: $b_{ii} = \sum_{k=1}^{n} a_{ik} \cdot a_{ki}$ reprezintă numărul orașelor k conectate direct cu orașul i ($a_{ik} = 1$) - 1p)

Problema 3 ..... **25 pct.**

3a ..... 15 pct.

(definirea unei structuri de date adecvate - 5p; declararea corectă a funcției/ procedurii și a variabilelor utilizate - 1p; parcurgerea tabloului cu dimensiuni și completarea structurii - 9p)

3b ..... 10 pct.

(justificarea faptului că un subșir strict crescător nu poate avea mai mult de un element în aceeași stivă - 2p; declararea corectă a funcției/procedurii și a variabilelor utilizate - 1p; construirea unui subșir strict crescător - 7p)

3.4 Sesiunea Iulie 2019

1. (35p)

   (a) (10p)

      i. (5p) Scrieți o funcție în C/C++/Pascal care returnează 1 dacă un număr întreg pozitiv primit ca parametru este prim și 0 în caz contrar.

      ii. (5p) Se consideră un tablou x cu n numere întregi pozitive. Scrieți o funcție/procedură în C/C++/Pascal care afișează toate perechile de indici (i, j) $(i \leq i < j \leq n)$ pentru care suma $x[i] + x[j]$ este număr prim (se va folosi funcția de la punctul precedent).

   (b) (10p) Se consideră o funcție $f : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, n\}$, $(m, n \in \mathbb{N}, m \geq n)$ specificată prin tabloul valorilor sale: v este un tablou unidimensional cu m elemente în care $v[i] = f[i]$ (se presupune că indicele tabloului pornește de la 1).

      i. (2.5p) Care dintre următoarele proprietăți este suficientă pentru a garanta faptul că funcția f este bijectivă?

         A. $m = n$;

         B. $m = n$ **ȘI** v are elemente distincte ($v[i] \neq v[j]$ pentru orice $i \neq j$);

         C. $m = n$ **SAU** v are elemente distincte ($v[i] \neq v[j]$ pentru orice $i \neq j$);

      ii. (2.5p) Câte funcții bijective pot fi definite pe o mulțime $\{1, 2, \ldots, m\}$?

         A. m;

         B. m^2;

         C. $m!$;

         D. C_m^2;

         E. A_m^2.

      iii. (5p) Scrieți o funcție în C/C++/Pascal care primește ca parametru un număr natural $m \geq 2$ și un tablou v cu m elemente și care returnează 1 dacă v are toate elementele distincte și 0 în caz contrar.

   (c) (10p) O secvență de cifre binare este considerată "complementar-concatenată" dacă poate fi construită astfel: se pornește de la cifra 0 și la fiecare etapă, secvența curentă este concatenată cu complementara ei. Exemple de astfel de secvențe sunt:

Număr de ordine	**Secvența**
1:	0
2:	01
3:	0110
4:	01101001
5:	0110100110010110
...	...

i. (1p) Care dintre afirmațiile referitoare la o secvență "complementar-concatenată" este adevărată?
   A. numărul de elemente egale cu 1 din secvența cu numărul de ordine n ($n \geq 2$) este $n-1$;
   B. numărul de elemente egale cu 0 din secvența cu numărul de ordine n ($n \geq 2$) este $2(n-2)$;
   C. numărul de elemente egale cu 1 din secvența cu numărul de ordine n ($n \geq 2$) este 2^{n-1};
   D. numărul de elemente egale cu 1 din secvența cu numărul de ordine n ($n \geq 2$) este 2^{n-2}.

ii. (4p) Scrieți o funcție în C/C++/Pascal care primește ca parametri indicii i și j a două elemente dintr-un tablou (variabilă globală) ce conține valori din $\{0, 1\}$ (astfel încât numărul de elemente din subtabloul definit de i și j este un număr par) și returnează 1 dacă cele două jumătăți ale subtabloului definit de i și j sunt complementare și 0 în caz contrar (de exemplu, în tabloul $(0, 0, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{0}, 1, 0)$) subtabloul specificat prin indicii $i = 3$ și $j = 10$ satisface proprietatea, dar subtabloul corespunzător din $(0, 0, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{1}, 1, 0)$ nu satisface proprietatea.

iii. (5p) Scrieți o funcție recursivă care verifică dacă un tablou este o secvență "complementar-concatenată". Tabloul va fi considerat variabilă globală (cu n elemente, unde n este o putere a lui 2), iar funcția va primi ca parametri doi indici care specifică porțiunea din tablou care va fi analizată în cadrul apelului curent.

(d) (5p) Se consideră că asupra unui tablou unidimensional a cu $2 \leq n \leq 100$ elemente din mulțimea $\{1, 2, \ldots, n\}$ inițial definit astfel încât $a[i] = i$ (se presupune că indicii încep de la 1) a fost aplicată o transformare prin deplasare circulară către dreapta sau stânga cu k poziții ($1 \leq k < n$). Se pune problema determinării lui k pornind de la tabloul transformat fără a efectua comparații între elementele tabloului.

i. (2.5p) Care dintre următoarele afirmații sunt adevărate?
   A. dacă deplasarea este la dreapta atunci $k = n - x[n]$
   B. dacă deplasarea este la stânga atunci $k = n - x[n]$
   C. dacă deplasarea este la dreapta atunci $k = x[1] - 1$
   D. dacă deplasarea este la stânga atunci $k = x[1] - 1$
   E. indiferent de sensul deplasării $k = n - x[n]$
   F. indiferent de sensul deplasării $k = x[1] - 1$
   G. nu se poate determina k fără a efectua comparații între elementele tabloului

ii. (2.5p) Dacă se cunoaște doar tabloul transformat, este posibil să se determine sensul deplasării? Argumentați.

2. (25p) Se consideră o expresie algebrică ce poate conține ca operanzi doar simbolurile x, y și z, iar ca operatori doar $+$ și $-$. Se presupune că expresia începe întotdeauna cu un operand (de exemplu, $x + y - z + y + z + y$ și $z + x + y - x$ sunt expresii valide, dar $-z + x + y - x$, $zx + y - x$, $z + -y - x$ nu sunt valide). Se presupune că expresia este reprezentată printr-un tablou unidimensional (E)

cu elemente de tip caracter. Să se scrie câte o funcție/procedură care primește tabloul de caractere, precum și numărul de elemente din acesta (N) și efectuează prelucrărilre:

(a) (10p) Returnează 1 dacă expresia este validă și 0 în caz contrar.

(b) (15p) Presupunând că expresia este validă, afișează varianta simplificată a expresiei (de exemplu, pentru $x + y - x + y + z + y$ varianta simplificată este $x + 3y$, iar pentru $y + x - z + x$ este $2x + y - z$).

Pentru fiecare dintre funcții, pe lângă implementarea în C/C++/Pascal, se va descrie pe scurt ideea de rezolvare.

3. (30p) Se consideră că o imagine alb-negru este reprezentată printr-o matrice I cu M linii și N coloane și elemente din $\{0, 1\}$ (0 corespunde cu alb și 1 corespunde cu negru) și se presupune că imaginea conține un disc alb pe fond negru, astfel încât discul este încadrat de cel puțin o linie și o coloană de elemente egale cu 1 (vezi Figura 1a).

   (a) (10p) Scrieți o funcție cu numele `Verific` care primește ca parametru un tablou unidimensional cu elemente din $\{0, 1\}$ și numărul de elemente din tablou și returnează: 0 dacă toate elementele din tablou sunt egale cu 1; 1 dacă tabloul este de forma $\{1, \ldots, 1, 0, \ldots, 0, 1, \ldots, 1\}$ (cel puțin un zero încadrat de cel puțin un unu); 2 dacă tabloul conține cel puțin două porțiuni de elemente egale cu 0 separate prin cel puțin un element egal cu 1 (de exemplu, $(1, 1, 0, 0, 1, 0, 1)$ și $(1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1)$ au această proprietate);

   (b) (10p) Scrieți o funcție care are acces la matricea I (considerată variabilă globală) și folosește funcția `Verific` descrisă la punctul (a) pentru a determina diametrul discului exprimat în număr de linii ale matricii (vezi Figura 3.1).

   (c) (10p) Se presupune că la achiziția imaginii ar fi putut interveni erori astfel că interiorul discului alb poate conține pete negre (vezi Figura 3.2). Prezentați pe scurt ideea unei metode care să permită verificarea prezenței unor pete negre în interiorul discului. Implementați ideea într-o funcție care are acces la matricea I și returnează 1 dacă discul nu conține pete negre și 0 în caz contrar.

   Indicație: se poate folosi funcția de la punctul (a).

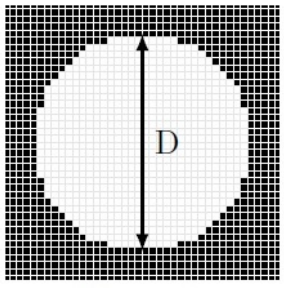


Figura 3.1

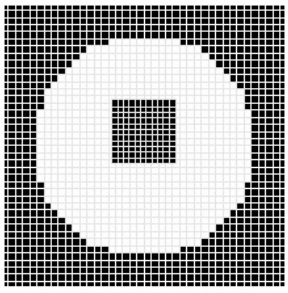


Figura 3.2

Notă: Pentru fiecare dintre funcțiile/procedurile scrise se vor specifica toate declarațiile de variabile globale și locale necesare și se vor pune comentarii explicative. **NU** este necesară scrierea programului principal și preluarea datelor prin citire.

BAREM

Start ..... **10 pct.**

Problema 1 ..... **35 pct.**

1a ..... 10 pct.

1(a)i. ..... 5 pct.

(declararea corectă a funcției, a variabilelor de lucru și returnarea rezultatului - 1p; descrierea corectă a algoritmului de verificare dacă un număr este prim - 4p)

1(a)ii. ..... 5 pct.

(declararea corectă a funcției/procedurii și a variabilelor de lucru - 1p; descrierea corectă a algoritmului de determinare a perechilor care au suma număr prim - 3p; afișarea perechilor - 1p)

1b ..... 10 pct.

1(b)i. ..... 2.5 pct.

(Răspuns corect: B.)

1(b)ii. ..... 2.5 pct.

(Răspuns corect: C.)

1(b)iii. ..... 5 pct.

(declararea corectă a funcției, a variabilelor de lucru și returnarea rezultatului - 1p; descrierea corectă a algoritmului de verificare dacă elementele vectorului sunt distincte - 4p)

1c ..... 10 pct.

1(c)i. ..... 1 pct.

(Răspuns corect: D.)

1(c)ii. ..... 4 pct.

(declararea corectă a funcției, a variabilelor de lucru și returnarea rezultatului - 1p; descrierea corectă a algoritmului de verificare dacă jumătățile tabloului sunt complementare - 3p)

1(c)iii. ..... 5 pct.

(declararea corectă a funcției, a variabilelor de lucru și returnarea rezultatului - 1p; condiție corectă pentru ieșirea din apelul recursiv - 1p; apel recursiv corect - 1p; verificarea proprietății de tablou complementar - 2p)

1d ..... 5 pct.

1(d)i. ..... 2.5 pct.

(Răspunsuri corecte: A., D.)

1(d)ii. ..... 2.5 pct.

(Nu este posibil să se determine sensul transformării deoarece același rezultat poate fi obținut prin deplasarea la dreapta cu k poziții și prin deplasarea la stânga cu h poziții. De exemplu, pentru $n = 5$, tabloul $(3, 4, 5, 1, 2)$ se poate obține atât prin deplasare la dreapta cu $k = 3$ poziții cât și prin deplasare la stânga cu $h = 2$ poziții).

Problema 2 ..... **25 pct.**

2a ..... 10 pct.

(declararea corectă a funcției, a variabilelor de lucru și returnarea rezultatului - 1p; descrierea corectă a algoritmului de verificare dacă expresia este validă - 7p; descriere idee de rezolvare - 2p)

2b ..... 15 pct.

(declararea corectă a funcției/procedurii și a variabilelor de lucru - 1p; determinarea corectă a coeficienților operanzilor (x, y și z) - 6p; afișarea corectă a expresiei simplificate: verificare existență operand - 3p; analiză semn - 3p; descriere idee de rezolvare - 2p)

Problema 3 ..... **30 pct.**

3a ..... 10 pct.

(declararea corectă a funcției și a variabilelor de lucru - 1p; descrierea corectă a algoritmului de verificare că toate elementele din tablou sunt egale cu 1 - 3p; descrierea corectă a algoritmului de verificare că există cel puțin un zero încadrat de 1 - 5p; returnarea rezultatului - 1p)

3b ..... 10 pct.

(declararea corectă a funcției și a variabilelor de lucru - 1p; descrierea corectă a algoritmului de determinare a indicilor liniilor de unde începe și unde se termină discul (de exemplu, linia i_1 și linia i_2) - 8p; calculul diametrului (de exemplu, $i_2 - i_1 + 1$) și returnarea rezultatului - 1p)

3c ..... 10 pct.

(descriere idee de rezolvare - 2p; declararea corectă a funcției, a variabilelor de lucru și returnarea rezultatului - 1p; descrierea corectă a algoritmului de verificare dacă discul conține sau nu pete, prin parcurgerea liniilor matricii, apelând funcția de la punctul (a) - 7p)