1. Se consideră un graf neorientat cu **5** noduri .Care dintre următoarele şiruri de

numere pot fi gradele nodurilor grafului? Precizati numarul de muchii. **(4p.)**

**a. 4**, **2**, **6**, **4**, **2 b. 2**, **2**, **1**, **2**, **2**

**c. 1**, **1**, **1**, **1**, **1 d. 4**, **3**, **3**, **4**, **4**

**2.** Scrieţi un program **C/C++** care citeşte de la tastatură o frază având maximum **100** de

caractere, în care cuvintele sunt separate prin câte un spaţiu; programul construieşte în

memorie şi afişează pe ecran un şir ce conţine **doar primul** caracter al fiecăruia dintre

cuvintele frazei, în ordinea în care acestea apar în frază, ca în exemplu.

**Exemplu:** dacă se citeşte fraza

**Ana sustine bacalaureatul la informatica**

atunci se va afişa **Asbli**

1. Se consideră mulţimea vocalelor **{a,e,i,o,u}**. Scrieţi o expresie **C/C++** care să fie

nenulă dacă şi numai dacă variabila **c** de tip **char** este o vocală.

1. Scrieţi programul **C/C++** care citeşte de la tastatură un cuvânt format din cel mult **50** de

caractere, litere mari ale alfabetului englez, şi afişează pe ecran, fiecare pe câte o linie,

toate prefixele acestuia, în ordine crescătoare a lungimilor. Un prefix de lungime **k** al unui

cuvânt este un subşir format din primele **k** caractere ale acestuia. **(10p.)**

**Exemplu**: dacă se citeşte cuvântul **BACALAUREAT** se vor afişa prefixele:

**B**

**BA**

**BAC**

**BACA**

**BACAL**

**BACALA**

**BACALAU**

**BACALAUR**

**BACALAURE**

**BACALAUREA**

**BACALAUREAT**

1. Scrieţi programul **C/C++** care citeşte de la tastatură un cuvânt format din cel mult **50**

caractere, litere mari ale alfabetului englez, şi afişează pe ecran, fiecare pe câte o linie,

toate sufixele acestuia, în ordine crescătoare a lungimilor. Un sufix de lungime **k** al unui

cuvânt este un subşir format din ultimele **k** caractere ale acestuia. **(10p.)**

**Exemplu**: dacă se citeşte cuvântul **EXAMEN** se vor afişa sufixele :

**N**

**EN**

**MEN**

**AMEN**

**XAMEN**

**EXAMEN**

1. Scrieţi programul **C/C++** care citeşte de la tastatură o valoare naturală **n** (**2≤n≤100)** şi

construieşte în memorie, apoi afişează pe ecran o matrice **a** cu **n** linii şi **n** coloane, simetrică

faţă de diagonala secundară. Elementele matricei sunt numerele naturale de la **1** la **n(n** + **1)/2**.

Elementele situate deasupra şi pe diagonala secundară sunt dispuse în ordine crescătoare pe linii astfel: prima linie conţine numerele de la **1** la **n**, a doua linie conţine numerele de la **n + 1** la **2\*n – 1** şi aşa mai departe. Matricea se va afişa pe ecran, câte o linie a matricei pe o linie a ecranului, elementele unei linii fiind separate între ele printr-un spaţiu.

**Exemplu:** pentru **n = 4** se va obţine matricea alăturată. **(10p.)**

**1 2 3 4**

**5 6 7 3**

**8 9 6 2**

**10 8 5 1**

1. Fişierul text **BAC.TXT** conţine pe prima linie un număr natural nenul **n** (**1**≤**n**≤**1000**), iar pe

fiecare dintre următoarele **n** linii câte două numere întregi **a** şi **b** (**1**≤**a**≤**b**≤**32000**), fiecare

pereche reprezentând un interval închis de forma **[a,b]**. Scrieţi un program **C/C++** care

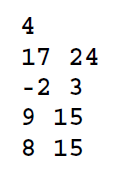
citeşte numerele din fişier şi determină un interval dintre cele citite care conţine cel mai

mare număr de numere întregi şi afişează pe o linie a ecranului, separate printr-un spaţiu,

numerele care reprezintă capetele intervalului determinat. În cazul în care sunt mai multe

intervale care îndeplinesc această proprietate, se vor afişa informaţiile referitoare la acel

interval la care numărul care reprezintă capătul din dreapta este minim.



**Exemplu**: dacă fişierul **BAC.TXT** are conţinutul alăturat, pe ecran se va

afişa: **8 15** (pentru că numărul maxim de numere întregi aflate într-un

interval este **8**. Sunt două intervale care conţin **8** numere: **[17;24]** şi

**[8;15]**. Dintre acestea, **[8;15]** are capătul din dreapta cel mai mic).

1. Scrieţi definiţa completă a subprogramului **nreal** cu doi parametri **x** şi **y**, numere naturale

din intervalul **[1;1000]** ce returnează un număr real cu proprietatea că partea sa întreagă

este egală cu **x**, iar cifrele numărului **y** sunt egale, în ordine, cu cifrele aflate după punctul

zecimal.

**Exemplu:** pentru **x=12** şi **y=543**, subprogramul returnează valoarea **12.543**.

1. Fişierul text **NUMERE.IN** conţine pe prima linie un număr natural nenul **n** (**2≤n≤100**) şi pe

următoarea linie **n** numere reale pozitive, în ordine strict crescătoare, separate prin câte un

spaţiu.

**a)** Scrieţi un program **C/C++** care, utilizând un algoritm eficient din punct de vedere al

memoriei utilizate, determină şi afişează pe ecran, separate printr-un spaţiu, două numere

naturale **x** şi **y** cu proprietatea că toate cele **n** numere aflate pe linia a doua în fişierul

**NUMERE.IN** se găsesc în intervalul **[x;y]** şi diferenţa **y-x** este minimă.

**Exemplu:** dacă fişierul **NUMERE.IN** are conţinutul:

**6**

**3.5 5.1 9.2 16 20.33 100** atunci se afişează **3 100 (6p.)**

**b)** Descrieţi în limbaj natural metoda utilizată şi explicaţi în ce constă eficienţa ei. **(4p.)**

1. Fişierul text **NUMERE.IN** conţine pe prima linie un număr natural nenul **n** (**1≤n≤100**) şi pe

următoarea linie **n** numere reale pozitive **ordonate crescător**, separate prin câte un spaţiu.

**a)** Scrieţi un program **C/C++** care citeşte din fişierul **NUMERE.IN** numărul natural **n**, şi

determină, utilizând un algoritm eficient din punct de vedere al timpului de executare şi al

memoriei utilizate, numărul de valori reale distincte de pe linia a doua din fişier.

**Exemplu:** dacă fişierul **NUMERE.IN** are conţinutul:

**6**

**2.3 2.3 2.8 5.7 5.7 6.3**

atunci se afişează valoarea **4** (sunt **4** valori distincte: **2.3, 2.8, 5.7, 6.3**). **(6p.)**

**b)** Descrieţi în limbaj natural metoda utilizată şi explicaţi în ce constă eficienţa ei.

**11.** Scrieţi în limbajul **C/C++** definiţia completă a subprogramului **ordonare** care primeşte ca

parametru un tablou unidimensional **x** cu cel mult **100** de elemente, numere naturale cu cel

mult **4** cifre fiecare, şi un număr natural **n** (**n≤100**), ce reprezintă numărul efectiv de

elemente ale tabloului **x**. Subprogramul va afişa tabloul obţinut în urma schimbarii poziţiei

doar a elementelor impare din tablou astfel încât acestea să apară în ordinea crescătoare a

valorilor lor. **(10p.)**

**Exemplu:** pentru **n=6** şi **x=(7,11,2,-8,-3,10)**

subprogramul va afişa **-3,7,2,-8,11,10**

**12.** În fişierul **numere.txt**, se află memorate, pe prima linie un număr natural **n** (**1**<=**n**<=**100**), iar

pe fiecare dintre următoarele **n** linii, câte două numere întregi **x,y** (**-100**<=**x<=y<=100**),

reprezentând capetele câte unui segment **[x,y]** desenat pe axa **Ox** de coordonate.

**a)** Scrieţi în limbajul **C/C++** un program eficient din punct de vedere al timpului de executare

şi al spaţiului de memorare, care citeşte din fişier datele existente, determină segmentul

rezultat în urma intersecţiei tuturor celor **n** segmente date şi afişează pe ecran două numere

despărţie printr-un spaţiu ce reprezintă capetele segmentului cerut. Dacă segmentele nu au

niciun punct comun se va afişa pe ecran valoarea **0**. **(6p.)**

**b)** Descrieţi în limbaj natural algoritmul utilizat, justificând eficienţa acestuia. **(4p.)**

**Exemplu:** dacă fişierul **numere.txt** are conţinutul alăturat, se va afişa

pe ecran

**3 5**

**5**

**-7 10**

**3 20**

**-5 5**

**0 12**

**-8 30**

**13.** Scrieţi în limbajul **C/C++** definiţia completă a subprogramului **ordonare** care primeşte ca

parametru un tablou unidimensional **x** cu cel mult **100** de elemente, numere naturale cu cel

mult **4** cifre fiecare, şi un număr natural **n** (**n≤100**), ce reprezintă numărul efectiv de

elemente ale tabloului **x**. Subprogramul va afişa tabloul obţinut în urma schimbarii poziţiei

doar a elementelor impare din tablou astfel încât acestea să apară în ordinea crescătoare a

valorilor lor. **(10p.)**

**Exemplu:** pentru **n=6** şi **x=(7,11,2,-8,-3,10)**

subprogramul va afişa **-3,7,2,-8,11,10**

**14.** În fişierul **numere.txt**, se află memorate, pe prima linie un număr natural **n** (**1**≤**n**≤**100**), iar

pe fiecare dintre următoarele **n** linii, câte două numere întregi **x,y** (**-100**≤**x**≤**y**≤**100**),

reprezentând capetele câte unui segment **[x,y]** desenat pe axa **Ox** de coordonate.

**a)** Scrieţi în limbajul **C/C++** un program eficient din punct de vedere al timpului de executare

şi al spaţiului de memorare, care citeşte din fişier datele existente, determină segmentul

rezultat în urma intersecţiei tuturor celor **n** segmente date şi afişează pe ecran două numere

despărţie printr-un spaţiu ce reprezintă capetele segmentului cerut. Dacă segmentele nu au

niciun punct comun se va afişa pe ecran valoarea **0**. **(6p.)**

**b)** Descrieţi în limbaj natural algoritmul utilizat, justificând eficienţa acestuia. **(4p.)**

**Exemplu:** dacă fişierul **numere.txt** are conţinutul alăturat, se va afişa

pe ecran

**3 5**

**5**

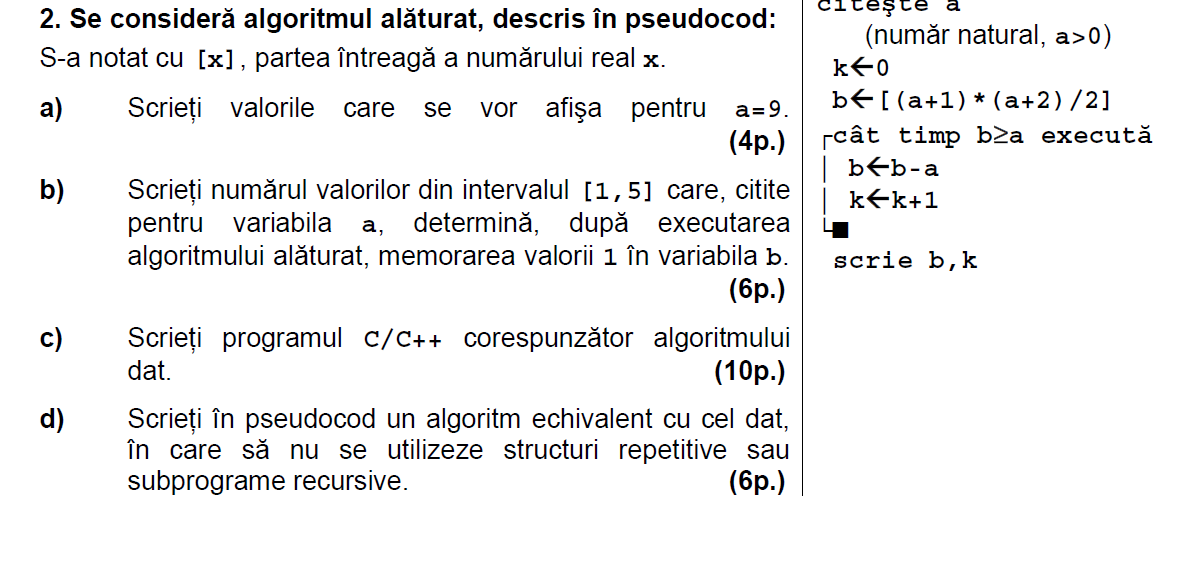
**-7 10**

**3 20**

**-5 5**

**0 12**

**-8 30**

****