SUBIECTUL I **(20 de puncte)**

Pentru fiecare dintre itemii de la 1 la 5, scrieţi pe foaia de examen litera corespunzătoare răspunsului corect. Fiecare răspuns corect se notează cu 4 puncte.

1. Variabila întreagă **n** memorează un număr natural. Indicați expresia C/C++ care are valoarea **1** dacă şi numai dacă numărul memorat în **n** este divizibil cu **20**, dar **NU** şi cu **19**.

a. `n%380==0 && n/20==0`
b. `n%380!=0 || !(n%19==0)`
c. `n%20==0 && n/19==0`
d. `!(n%20!=0 || n%19==0)`

2. Subprogramul **f** este definit alăturat. Indicați apelul care determină afişarea, în ordine strict descrescătoare, a tuturor divizorilor proprii pozitivi ai numărului **1000** (divizori diferiți de **1** și de **1000**).

```
void f (int n, int d)
{ if(d<n/2) f(n, d+1);
  if(n%d==0)
    cout<<d<<' '; | printf("%d ",d);
}
```

a. `f(1000,2);`
b. `f(999,2);`
c. `f(500,2);`
d. `f(32,2);`

3. Utilizând metoda backtracking, se generează toate parfumurile formate prin amestecarea a câte **3** esențe distincte din mulţimea {**agar**, **geranium**, **iasomie**, **paciuli**, **tuberoze**}. Primele patru soluţii obţinute sunt, în această ordine: (**agar**, **geranium**, **iasomie**), (**agar**, **geranium**, **paciuli**), (**agar**, **geranium**, **tuberoze**) şi (**agar**, **iasomie**, **paciuli**). Indicaţi soluția generată imediat înainte de (**geranium**, **iasomie**, **paciuli**).

a. (**agar**, **iasomie**, **paciuli**)
b. (**agar**, **paciuli**, **tuberoze**)
c. (**geranium**, **paciuli**, **iasomie**)
d. (**geranium**, **agar**, **iasomie**)

4. Un arbore cu **10** noduri, numerotate de la **1** la **10**, este reprezentat prin vectorul de „taţi” **(6,5,7,5,9,9,6,7,0,5)**. Numărul nodurilor de tip “frunză” ale arborelui este:

a. 4
b. 5
c. 6
d. 7

5. Un graf neorientat are **10** muchii și este conex. Numărul maxim de noduri ale sale este:

a. 8
b. 9
c. 10
d. 11

SUBIECTUL al II-lea **(40 de puncte)**

1. Se consideră algoritmul alăturat, reprezentat în pseudocod.

```
citește n
    (număr natural)
 k←1
┌cât timp n≥1 execută
│┌dacă n>k atunci i←k
│|altfel i←n
│└■
│ n←n-i
│┌cât timp i≥1 execută
│| scrie k,' '; i←i-1
│└■
│ k←k+1
└■
```

a) Scrieţi valorile afişate dacă se citește numărul **7**. **(6p.)**

1 2 2 3 3 3 4

b) Scrieţi cel mai mic și cel mai mare număr care pot fi citite astfel încât, în urma executării algoritmului, pentru fiecare dintre acestea, ultima valoare afişată să fie **10**. 46,55 **(6p.)**

c) Scrieţi programul C/C++ corespunzător algoritmului dat. **(10p.)**

d) Scrieţi în pseudocod un algoritm, echivalent cu cel dat, înlocuind una dintre structurile **cât timp...execută** cu o structură repetitivă de alt tip. **(6p.)**

c) si d)

```
#include <iostream>
using namespace std;
int main()
{
unsigned long n, k,i;
cin>>n;
k=1;
while(n>=1)
{
        if(n>k)i=k;
                else i=n;
        n=n-i;
        while(i>=1)
                {cout<<k<<' ';i=i-1;}
        k=k+1;
}
return 0;
}
```

```
citește n
    (număr natural)
 k←1
┌cât timp n≥1 execută
│┌dacă n>k atunci i←k
│|altfel i←n
│└■
│ n←n-i
│┌cât timp i≥1 execută
│|
│└■
│ k←k+1
└■
```

```
repeta
 scrie k,' '; i←i-1
pana cand i<1
```

2. Pentru un număr complex se memorează următoarele date: partea reală și partea imaginară (numere reale). Variabila **z** memorează simultan date pentru fiecare dintre cele **20** de numere complexe. Știind că expresia C/C++ de mai jos are valoarea sumei dintre partea reală și partea imaginară ale primului număr complex dintre cele precizate, scrieți definiția unei structuri cu eticheta **complex** care să permită memorarea datelor unui număr complex, și declarați corespunzător variabila **z**.

z[0].pre+z[0].pim

```
struct complex{
float pre,pim;
}z[20];
```

(6p.)

3. Variabilele **i** şi **j** sunt de tip întreg, iar variabila **a** memorează un tablou bidimensional cu **5** linii şi **5** coloane, numerotate de la **1** la **5**, având iniţial toate elementele nule.
Fără a utiliza alte variabile decât cele menţionate, scrieţi secvenţa de instrucţiuni de mai jos, înlocuind punctele de suspensie astfel încât, în urma executării secvenţei obţinute, variabila **a** să memoreze tabloul alăturat.

```
for(i=1;i<=5;i++)
  for(j=1;j<=5;j++)
     a[i][j]=(i-1)*5+j;
```

(6p.)

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

SUBIECTUL al III-lea (30 de puncte)

1. Subprogramul **CifrePrime** are un singur parametru, **n**, prin care primeşte un număr natural (**$n \in [0, 10^9]$**). Subprogramul returnează suma cifrelor prime ale lui **n**.
Scrieţi definiţia completă a subprogramului.
Exemplu: dacă **n=1235405**, atunci subprogramul returnează **15**, iar dacă **n=140**, atunci subprogramul returnează **0**. **(10p.)**

```
int CifrePrime(unsigned long n)
{
int s=0,r;
while(n!=0)
{
        r=n%10;
        n=n/10;
        if(r==2 || r==3 || r==5 || r==7) s=s+r;
}
return s;
}
```

2. Într-un text cu cel mult **100** de caractere, cuvintele sunt formate numai din litere mici ale alfabetului englez și sunt separate prin unul sau mai multe spații.
Scrieți un program C/C++ care citește de la tastatură un astfel de text, cu cel puțin trei cuvinte, și construiește în memorie un șir de caractere format din prima consoană a primului cuvânt, urmată de prima vocală a celui de al doilea cuvânt, respectiv de ultima literă a ultimului cuvânt, în ordinea în care acestea apar în text. Șirul obținut se afișează pe ecran, iar dacă nu se poate obține un astfel de șir, se afișează pe ecran mesajul **nu exista**. Se consideră vocale literele **a**, **e**, **i**, **o**, **u**.
Exemplu: pentru textul **el prefera sa mearga la schi**
se afișează pe ecran șirul **lei**
iar pentru textul **ei prefera sa mearga la schi**
se afișează pe ecran mesajul **nu exista** **(10p.)**

```
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;
int main()
{
  char str[101];
  char *p;
  int i,n; char v[50][20],sir[4];
```

```
   cin.get(str,100);
   n=0;
   p = strtok(str, " ");
   while( p != NULL )
   {
     n++; strcpy(v[n],p);
     p = strtok(NULL, " ");
   }
//for(i=1;i<=n;i++) cout<<v[i]<<' ';

   strcpy(sir,"");
//prima consoana a primului cuvant
   int gasitc=0;
   i=0;
   while(i<strlen(v[1]) && gasitc==0)
   {
    if(strchr("aeiou",v[1][i])==0) {sir[0]=v[1][i];gasitc=1;}
    i++;
   }
//prima vocala din al doilea cuvant
   int gasitv=0;
   i=0;
   while(i<strlen(v[2]) && gasitv==0)
   {
    if(strchr("aeiou",v[2][i])!=0) {sir[1]=v[2][i];gasitv=1;}
    i++;
   }
//ultima litera din ultimul cuvant
   sir[2]=v[n][strlen(v[n])-1];
   sir[3]='\0';
if(gasitv==1 && gasitc==1) cout<<sir;
  else cout<<"nu exista";
return 0; }
```

3. Un interval este numit **prieten de grad n** al unui șir dacă sunt exact **n** termeni ai șirului cu valori din interval și dacă toate numerele întregi care aparțin intervalului sunt valori ale unor termeni ai șirului. Fișierul **bac.txt** conține un șir de cel mult 10^6 numere naturale din intervalul $[0,10^2]$, separate prin câte un spațiu. Se cere să se afișeze pe ecran numărul maxim **n** cu proprietatea că există un interval prieten de grad **n** al șirului aflat în fișier. Proiectați un algoritm eficient din punctul de vedere al timpului de executare.

Exemplu: dacă fișierul conține numerele

10 10 11 3 4 2 49 4 2 3 21 2 27 12 13 14 15 5

atunci se afișează pe ecran **8** (intervalului **[2,5]** îi aparțin **8** termeni ai șirului)

a) Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**

b) Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream f("bac.txt");

int main()
{
int fr[101],x,i;
for(i=0;i<=100;i++) fr[i]=0;
while(f>>x)
        fr[x]++;
int s=0,n=0;
for(i=0;i<=100;i++)
{
        if(fr[i]!=0)s=s+fr[i];
```

```
        else
                {
                 if(s>n)n=s;
                 s=0;}
}
cout<<n;
return 0;
}
```

Variabilele folosite au rolul:
tabloul unidimensional fr – retine frecventa numerelor din {0, 1, 2,.., 100}
s – suma valorilor retinute in vectorul frecventa pentru secventa din vectorul fr curenta
n – numarul maxim cerut

Pentru fiecare numar din [0,100] vom retine frecventa de aparitie. Numarul n retine valoarea sumei maxime din secventele compuse din elemente consecutive de numere diferite de 0 din fr.

Programul foloseste putine variabile. Singurul vector folosit este un vector pentru a gasi frecventa de aparitie a unor valori; acest vector are un numar constant, mic de elemente. Prin urmare, programul nu foloseste multa memorie cand este executat.
Programul nu are instructiuni repetitive imbricate, prin urmare timpul necesar pentru executie este mic.