SUBIECTUL I **(20 de puncte)**

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

1. Indicați o expresie C/C++ care are valoarea 1 dacă și numai dacă numărul natural memorat în variabila întreagă **x** are exact o cifră.

a. `x/10==0` **b.** `x%10==0`

c. `(x/10)/10==0` **d.** `(x%10)%10==0`

2. Subprogramul **f** este definit alăturat. Scrieți ce se afișează în urma apelului de mai jos.
`f(3);`

```
void f(int n)
{ int i;
  for(i=n;i>=1;i--)
  { f(n-1);
    cout<<i;   |   printf("%d",i);
  }
}
```

a. 121321 **b.** 121131211212111 **c.** 322111 **d.** 321112211112111

3. Utilizând metoda backtracking, se generează toate modalitățile de a pregăti o ținută, luând, într-o anumită ordine, articolele din mulțimea {**cămașă**, **cravată**, **pantaloni**, **pantofi**, **sacou**, **șosete**}, având în vedere următoarele restricții: cămașa va fi luată înaintea cravatei, cravata înaintea sacoului și atât șosetele, cât și pantalonii, înaintea pantofilor. Primele trei soluții generate sunt, în această ordine: (**cămașă**, **cravată**, **pantaloni**, **sacou**, **șosete**, **pantofi**), (**cămașă**, **cravată**, **pantaloni**, **șosete**, **pantofi**, **sacou**), (**cămașă**, **cravată**, **pantaloni**, **șosete**, **sacou**, **pantofi**). Indicați cea de a șasea soluție generată.

a. (**cămașă**, **cravată**, **sacou**, **șosete**, **pantaloni**, **pantofi**)

b. (**cămașă**, **cravată**, **șosete**, **pantaloni**, **sacou**, **pantofi**)

c. (**cămașă**, **cravată**, **șosete**, **pantaloni**, **pantofi**, **sacou**)

d. (**cămașă**, **cravată**, **șosete**, **sacou**, **pantaloni**, **pantofi**)

4. Un arbore cu 9 noduri, numerotate de la 1 la 9, este reprezentat prin vectorul de „tați" (2,7,0,8,1,5,3,9,2). Rădăcina arborelui este:

a. 1 **b.** 3 **c.** 4 **d.** 6

5. Matricea de adiacență a unui graf neorientat cu 2020 de noduri are 200 de elemente nenule. Numărul maxim de componente conexe ale grafului este:

a. 2006 **b.** 2000 **c.** 1820 **d.** 400

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

1. **Algoritmul alăturat este reprezentat în pseudocod.**
S-a notat cu **a%b** restul împărțirii numărului natural **a** la numărul natural nenul **b** și cu **[c]** partea întreagă a numărului real **c**.

a. Scrieți ce se afișează dacă se citește numărul 100. D10 **(6p.)**

b. Scrieți toate numerele din intervalul [1,9] care pot fi citite astfel încât, pentru fiecare dintre acestea, în urma executării algoritmului, să se afișeze **N**. 2 3 5 6 7 8 **(6p.)**

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

d. Scrieți în pseudocod un algoritm echivalent cu cel dat, care să nu cuprindă nicio structură repetitivă. **(6p.)**

```
citește n
  (număr natural nenul)
x←1; y←n; d←2
┌cât timp x<y execută
│┌dacă n%d=0 atunci
│| x←d
│| y←[n/d]
│└■
│ d←d+1
└■
┌dacă x=y atunci
│ scrie 'D',x
│altfel scrie 'N'
└■
```

c)

```
#include <iostream>
using namespace std;
int main()
{
unsigned n,x,y,d;
cin>>n;
x=1;y=n;d=2;
while(x<y)
{
if(n%d==0)
   {x=d;y=n/d;}
d=d+1;
}
if(x==y) cout<<'D'<<x;
else cout<<'N';
return 0;
}
```

d)

```
citeste n (numar natural)
x← 0
daca radical(n)*radical(n)==n
        x=radical(n);
daca (x!=0) scrie 'D',x
   altfel scrie 'N'
```

2. Variabila **s** memorează simultan numărul de soluții complexe ale unei ecuații (număr natural din intervalul **[2,10²)**) și soluțiile propriu-zise (partea reală și partea imaginară, numere reale). Știind că expresiile C/C++ de mai jos au ca valori numărul de soluții ale unui ecuații, partea reală, respectiv partea imaginară a primei sale soluții, scrieți definiția unei structuri cu eticheta **ecuatie**, care permite memorarea datelor despre soluțiile unei ecuații, și declarați corespunzător variabila **s**.

s.numar **s.solutie[0].pre** **s.solutie[0].pim** **(6p.)**

```
struct sol{
   float pre,pim;
}
```

```
struct ecuatie{
   int numar;
   struct sol solutie[101];
}s;
```

3. Variabilele **i** și **j** sunt de tip întreg, iar variabila **a** memorează un tablou bidimensional cu **6** linii și **6** coloane, numerotate de la **0** la **5**, având inițial toate elementele egale cu caracterul @. Fără a utiliza alte variabile, scrieți secvența de instrucțiuni de mai jos, înlocuind punctele de suspensie astfel încât, în urma executării secvenței obținute, variabila **a** să memoreze tabloul alăturat. **(6p.)**

```
( ( ( ) ) )
* ( ( ) ) *
* * ( ) * *
* * ( ) * *
* ( ( ) ) *
( ( ( ) ) )
```

```
for(i=0;i<6;i++)
  for(j=0;j<6;j++)
      if(j<i && j<5-i || j>i && j>5-i) a[i][j]='*';
                  else
                      if(j<=2) a[i][j]='(';
                          else
                              a[i][j]=')';
```

SUBIECTUL al III-lea (30 de puncte)

1. Un număr este scris în baza de numerație **b** (**b≤10**) dacă cifrele sale aparțin intervalului **[0,b-1]**. Subprogramul **baza** are un singur parametru, **n**, prin care primește un număr natural (**n∈[0,10⁹]**). Subprogramul returnează cea mai mică bază din intervalul **[2,10]** căreia i-ar putea corespunde scrierea lui **n**. Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=50731**, subprogramul returnează numărul **8**. **(10p.)**

Trebuie gasita cifra de valoare maxima din numar; baza este mai mare cu 1 decat aceasta cifra;

```
int baza(unsigned long n)
{
int b=0;
while(n!=0)
{
        if(b<n%10) b=n%10;
        n=n/10;
}
return b+1;
}
```

2. Un text cu cel mult **100** de caractere conține cuvinte și numere, separate prin câte un spațiu. Cuvintele sunt formate numai din litere mici ale alfabetului englez, iar numerele sunt reale, pozitive, cu partea zecimală și partea întreagă separate prin simbolul virgulă, sau numai cu partea întreagă, ca în exemplu. Scrieți un program C/C++ care citește de la tastatură un text de tipul precizat și afișează pe ecran numărul de valori întregi din text.

Exemplu: pentru textul
grus leucogeranus are 1,40 m inaltime si traieste intre 30 si 40 de ani
se afișează pe ecran **2** **(10p.)**

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

```
  cin.get(str,100);
  n=0;
  p = strtok(str, " ");
  while( p != NULL )
  {
    n++; strcpy(v[n],p);
    p = strtok(NULL, " ");
  }

 int c=0;
 for(i=1;i<=n;i++)
   if((strchr("0123456789",v[i][0])!=0) && strchr(v[i],',')==0) c++;

//for(i=1;i<=n;i++) cout<<v[i]<<' ';
cout<<c;
return 0; }
```

3. Fișierul **bac.txt** conține un șir de cel mult 10^6 numere întregi din intervalul $[-10^3, 10^3]$, separate prin câte un spațiu. Se cere să se afișeze pe ecran suma maximă obținută adunând numere de pe poziții consecutive în șirul aflat în fișier. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate și al timpului de executare.
Exemplu: dacă fișierul **bac.txt** conține valorile **4 -6 7 2 -1 4 -10 -3 9 2 -2**
se afișează pe ecran numărul **12**

a. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**
b. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**

```
#include<iostream>
#include<fstream>
using namespace std;

ifstream f("bac.txt");

long s, i, smax, x, n;
int main()
{
f>>x;
smax=x;
s=x;
if(s<0)s=0;
while(f>>x)
{
s=s+x;
if(s>smax) smax=s;
if(s<0) s=0;
}
cout<<smax;
return 0;
}
```

Am notat cu:
smax- suma maxima a unei secvente din sirul de numere dat
s- suma secventei de valori curente pozitive
Sirul va fi impartit de secvente cat mai lungi de sume pozitive si negative. Sumele pozitive pot fii folosite pentru a calcula suma maxima, sumele negative nu vor fi folosite in calculul sumei. La fiecare pas vom adauga la s valoarea curenta si vom actualiza, dupa caz, valoarea sumei maxime. Daca suma s devine negativa vom reintializa valoarea acestei sume cu 0 si vom incepe calculul pentru o noua secventa.
Algoritmul afiseaza solutia corecta in cazul si in care sirul e format numai numai din numere negative, valaore maxima fiind egala cu valoarea elementului de valoare maxima din sir.
Algoritmul foloseste o singura structura repetitiva deci timpul de executie este mic.
Algoritmul nu foloseste tablouri; in algoritm am folosit putine variabile deci spatiul memorat de variabilele necesare pentru executia programului este mic.