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 două cifre.

a. `x/100!=0 || x/10==0` **b.** `x/100==0 && x/10!=0`

c. `x%100!=0 || x%10==0` **d.** `x%100==0 && x%10!=0`

2. Subprogramul **f** este definit alăturat. Scrieți valoarea lui **f(3)**.

```
int f(int n)
{ int r, i;
  r=0;
  for(i=1;i<=n;i++) r=r+i+f(n-i);
  return r;
}
```

a. 3 **b.** 6 **c.** 9 **d.** 11

3. Utilizând metoda backtracking, se generează toate modalitățile de a pregăti o masă festivă, servind, într-o anumită ordine, preparatele din mulțimea {**ciuperci cu spanac**, **legume la cuptor**, **pere umplute cu nucă**, **panna cotta de cocos cu vanilie**, **salată cu smochine**, **tort de lămâie**}, având în vedere următoarele restricții: ciupercile vor fi servite înaintea legumelor, legumele înaintea salatei și atât tortul, cât și perele umplute, înainte de panna cotta. Două soluții sunt distincte dacă ordinea servirii preparatelor este diferită. Primele trei soluții generate sunt, în această ordine: **(ciuperci cu spanac, legume la cuptor, pere umplute cu nucă, salată cu smochine, tort de lămâie, panna cotta de cocos cu vanilie)**, **(ciuperci cu spanac, legume la cuptor, pere umplute cu nucă, tort de lămâie, panna cotta de cocos cu vanilie, salată cu smochine)**, **(ciuperci cu spanac, legume la cuptor, pere umplute cu nucă, tort de lămâie, salată cu smochine, panna cotta de cocos cu vanilie)**. Indicați cea de a șasea soluție generată.

a. **(ciuperci cu spanac, legume la cuptor, salată cu smochine, tort de lămâie, pere umplute cu nucă, panna cotta de cocos cu vanilie)**

b. **(ciuperci cu spanac, legume la cuptor, tort de lămâie, pere umplute cu nucă, salată cu smochine, panna cotta de cocos cu vanilie)**

c. **(ciuperci cu spanac, legume la cuptor, tort de lămâie, pere umplute cu nucă, panna cotta de cocos cu vanilie, salată cu smochine)**

d. **(ciuperci cu spanac, legume la cuptor, tort de lămâie, salată cu smochine, pere umplute cu nucă, panna cotta de cocos cu vanilie)**

4. Un arbore cu **9** noduri, numerotate de la **1** la **9**, este reprezentat prin vectorul de „tați" **(8, 7, 0, 6, 2, 8, 3, 3, 2)**. Indicați rădăcina arborelui.

a. 1 **b.** 3 **c.** 5 **d.** 9

5. Matricea de adiacență a unui graf neorientat cu **100** de noduri are **9900** de elemente nule. Indicați numărul maxim de componente conexe ale grafului.

a. 50 **b.** 90 **c.** 1000 **d.** 9800

3.
Notam:
1-ciuperci cu spanac
2-legume la cuptor
3-pere umplute cu nucă
4-panna cotta de cocos cu vanilie
5-salată cu smochine
6-tort de lămâie

Prin urmare, restrictiile de servire sunt:
12
25
64
34

Primele 6 solutii sunt:
1 2 3 5 6 4
1 2 3 6 4 5
1 2 3 6 5 4
1 2 5 3 6 4
1 2 5 6 3 4
1 2 6 3 4 5

4. Radacina corespunde pozitiei unde se afla 0 in vectorul de tati

5. O matrice de adiacenta pentru 100 noduri are 10000 elemente; 9900 elemente sunt nule, prin urmare, 100 elemente au valoarea 1; matricea de adiacenta in cazul grafurilor neorientate este simetrica fata de diagonala principala.
100 elemente cu valoarea 1 in matricea de adiacenta corespund la 50 muchii.

Cel mai mic numar n cu proprietatea ca n(n-1)/2>=50 => se formeaza o componenta cu 11 noduri
Prin urmare, vom avea 89 componente conexe formate dintr-un singur nod si o componenta conexa formata din 11 noduri => in total 90 componente conexe

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`.

```
citește n
  (număr natural nenul)
d←1; s←0
┌cât timp d*d<n execută
│ ┌dacă n%d=0 și d%2≠[n/d]%2 atunci
│ │ s←s+d+[n/d]
│ └■
│ d←d+1
└■
┌dacă d*d=n atunci s←s+d
└■
scrie s
```

a. Scrieti ce se afisează dacă se citește numărul `12`. **(6p.)**
20

b. Scrieți două numere din intervalul $[2,10^2]$, unul par și unul impar, care pot fi citite astfel încât, pentru fiecare dintre acestea, în urma executării algoritmului, să se afișeze `9`. 8, 81 **(6p.)**

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

d. Scrieți în pseudocod un algoritm, echivalent cu cel dat, înlocuind adecvat structura `cât timp...execută` cu o structură repetitivă de alt tip. **(6p.)**

c)

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

int main()
{
unsigned int n,d,s;
cin>>n;
d=1; s=0;
while(d*d<n)
{
        if(n%d==0 && d%2!=(n/d)%2)
        s=s+d+n/d;
        d=d+1;
}
if(d*d==n) s=s+d;
cout<<s;
return 0;}
```

d)

```
citește n
  (număr natural nenul)
d←1; s←0
repeta
┌dacă n%d=0 și d%2≠[n/d]%2 atunci
│ s←s+d+[n/d]
└■
d←d+1
pana cand d*d>=n
┌dacă d*d=n atunci s←s+d
└■
scrie s
```

2. Variabila `p` memorează simultan date specifice pentru prepararea unei prăjituri: numărul de ingrediente necesare și, pentru fiecare ingredient, codul acestuia și cantitatea necesară. Știind că expresiile C/C++ de mai jos au ca valori numere naturale din intervalul `[2,20]`, reprezentând numărul de ingrediente necesare pentru prepararea unei prăjituri, codul primului ingredient și cantitatea necesară din acesta, scrieți definiția unei structuri cu eticheta `prajitura`, care permite memorarea datelor specifice pentru prepararea unei prăjituri, și declarați corespunzător variabila `p`.

`p.numar` `p.ingredient[0].cod` `p.ingredient[0].cantitate` **(6p.)**

```
struct INGREDIENT{
unsigned int cod, cantitate;
}
```

```
struct prajitura{
unsigned int numar;
struct INGREDIENT ingredient[21];
}p;
```

3. Variabilele `i` și `j` sunt de tip întreg, iar variabila `a` memorează un tablou bidimensional cu `7` linii și `7` coloane, numerotate de la `0` la `6`, 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.

```
for(i=0;i<7;i++)
  for(j=0;j<7;j++)
    ..................
```

(6p.)

b	a	a	a	a	a	b
b	b	a	a	a	b	b
b	b	b	a	b	b	b
b	b	b	b	b	b	b
b	b	b	a	b	b	b
b	b	a	a	a	b	b
b	a	a	a	a	a	b

```
for(i=0;i<7;i++)
        for(j=0;j<7;j++)
                if(i<4 && (j>i && j<6-i)) a[i][j]='a';
                else
                        if(i>=4 && (j<i && j>6-i)) a[i][j]='a';
                                else a[i][j]='b';
```

SUBIECTUL al III-lea (30 de puncte)

1. Subprogramul `transformareBaza10` are doi parametri, `b` și `n`, prin care primește câte un număr natural ($b\in[2,10]$, $n\in[0,10^9]$). Subprogramul returnează suma tuturor produselor de forma $c\cdot b^k$, unde `c` este cifra de pe poziția `k` în scrierea numărului `n`; pozițiile sunt numerotate de la dreapta la stânga, iar cifra unităților este pe poziția `0`.
Scrieți definiția completă a subprogramului.
Exemplu: dacă `b=2` și `n=10010`, subprogramul returnează `18` ($18=1\cdot2^4+0\cdot2^3+0\cdot2^2+1\cdot2^1+0\cdot2^0$). **(10p.)**

```
int baza(unsigned long n, int b)
{
          unsigned int s, p; int c;
s=0;
p=1;
while (n!=0)
{
c=n%10;
s=s+c*p;
p=p*b;
n=n/10;
}
return s;
}
```

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 întreagă și partea zecimală 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 îl transformă în memorie, înlocuind fiecare număr real cu partea întreagă a acestuia.
Exemplu: pentru textul
`partea intreaga a lui 5,75 este egala cu a lui 5,25 si cu a lui 5 si este 5`
se afișează pe ecran
`partea intreaga a lui 5 este egala cu a lui 5 si cu a lui 5 si este 5` **(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=strchr(v[i],',')-v[i];//pozitia virgulei in sir
         if(c!=0){
              strncpy(sir,v[i],c); //pastram subsirul din v[i] ce contine caracterele pana la virgula
              strcpy(v[i],sir);}
    }
for(i=1;i<=n;i++) cout<<v[i]<<' ';
return 0; }
```

3. Fișierul `bac.txt` conține un șir de cel mult 10^6 numere naturale din intervalul $[0, 10^3]$, separate prin câte un spațiu. Se cere să se afișeze pe ecran suma maximă obținută adunând numere cu aceeași paritate, aflate 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 `10 115 1 5 2 2 2 4 7 3 100 20 2 7`
se afișează pe ecran numărul `122`

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

Semnificatia variabilelor:
s - suma numerelor consecutive de aceasi paritate pana la pasul curent
smax - cea mai mare suma de valori consecutive de aceasi paritate
x- variabila ce retine ultima valoare citita din fisier
par – variabila care are valoarea 1 daca x este numar par si par=0 in cazul in care x este numar impar; paritatea ultimului element citit

Citim pe rand, din fisier, valorile intr-o singura variabila (x).
Pentru a retine paritatea ultimului numar a carui valoare s-a citit din fisier este nevoie sa citim o prima valoare din fisier; pana la acest punct, pentru x - prima valoare citita din fisier, vom initializa:
- valoarea variabilei par in functie de paritatea lui x
- suma numerelor cosecutive de aceasi paritate cu 0 (s=0)
- suma maxima obtinuta pentru elemente consecutive de aceasi paritate cu 0 (smax=0)
Vom citi restul de elemente pe rand; pentru fiecare element:
- daca inaintea lui se afla un element de aceasi paritate, adaugam la suma valoarea acestui element
- daca inaintea lui se afla un element de paritate diferita atunci reinitializam suma elementelor consecutive de aceasi paritate cu valoarea acestui element si schimbam paritatea ultimului citit in functie de paritatea lui x
Dupa ce am reinitializat valoarea sumei s verificam daca suma s astfel obtinuta are valoare mai mare decat cea mai mare suma de elemente consecutive de aceasi paritate si reinitializam pe smax dupa caz.

Programul foloseste putine variabile, prin urmare nu este nevoie sa se rezerve un memorie multa pentru variabile.
Programul contine o singura instructiune repetitiva ce se executa de n-1 ori (unde am notat cu n numarul de valori din bac.txt), deci, timpul de executie pentru program este mic.

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

ifstream f("bac.txt");

int s, i, smax, x, n,par;
int main()
{
f>>x;
smax=x;
s=x;
if(x%2==0) par=1;
            else par=0;

while(f>>x)
{
if(par==1 && x%2==0|| par==0 && x%2==1) s=s+x; //se adauga x la suma de numere pare/impare consecutive
  else
    if(par==0 && x%2==0)
          {
          //incepem sa construim suma de numere pare consecutive
          s=x;
          par=1;}
                    else
                      {//incepem sa construim suma de numere impare consecutive
                      s=x;
                      par=0;}

if(s>smax) smax=s;
}
cout<<smax;
return 0;
}
```