

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 care are valoarea 1 dacă și numai dacă numărul memorat în variabila întreagă x aparține intervalului $[-2002, 2020]$.

- a. $!(x < -2002 \mid \mid x > 2020)$ b. $x > -2002 \mid \mid x < 2020$
c. $!(x + 2020 < 0 \ \&\& \ x - 2002 > 0)$ d. $x + 2020 >= 0 \ \&\& \ x - 2002 <= 0$

2. Subprogramul f este definit alăturat. Valoarea lui $f(2020, 20)$ este:

```
int f(int x, int y)
{ if(x <= 0) return y;
  if(y <= 0) return x;
  return f(x/y, x-y);
}
```

- a. 0 b. -1 c. -1899 d. -2000

3. Utilizând metoda backtracking se generează toate grupele de accesorii pentru înot din mulțimea {cască, costum de înot, cleme, dopuri de urechi, plută, mască de snorkeling, ochelari, snorkel}. Accesoriile au prețurile următoare, exprimate în lei: cască - 40, costum de înot - 400, cleme - 30, dopuri de urechi - 30, plută - 10, mască de snorkeling - 400, ochelari - 70, snorkel - 60. Într-o grupă accesoriiile sunt distincte, nu contează ordinea lor și costă, în total, exact 500 de lei. Primele trei soluții generate sunt, în această ordine: (cască, costum de înot, cleme, dopuri de urechi), (cască, costum de înot, snorkel), (cască, cleme, dopuri de urechi, mască de snorkeling). A cincea soluție generată este:

- a. (cască, mască de snorkeling, snorkel)
b. (costum de înot, cleme, plută, snorkel)
c. (cleme, mască de snorkeling, ochelari)
d. (dopuri de urechi, plută, mască de snorkeling, snorkel)

4. Într-un arbore cu rădăcină cu 6 noduri, numerotate de la 1 la 6, sunt 3 noduri de tip „frunză”. Indicați un vector de „tați” care poate corespunde arborelui.

- a. (0, 3, 1, 6, 6, 2) b. (2, 0, 1, 1, 2, 1) c. (3, 5, 5, 6, 0, 5) d. (5, 5, 1, 1, 0, 1)

5. Un graf orientat cu 6 vârfuri, numerotate de la 1 la 6, este reprezentat prin matricea de adiacență alăturată. Indicați numărul minim de arce care pot fi adăugate astfel încât graful obținut să aibă cel puțin două circuite. Un circuit este format numai din arce distincte, iar două circuite sunt distincte dacă diferă prin cel puțin un arc.

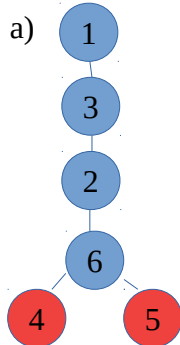
0	0	0	0	0	1
1	0	1	0	1	0
0	0	0	1	1	0
0	0	0	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0

- a. 1 b. 2 c. 3 d. 4

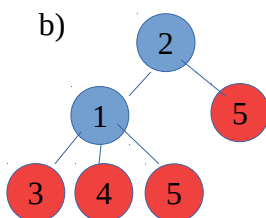
3)

$f(2020, 20) =$	$f(101, 2000)$	1899
$f(101, 2000) =$	$F(0, -1899)$	-1899
$F(0, -1899) =$	$-1899 \ (x \leq 0)$	

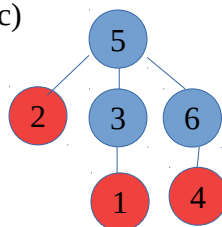
4) a)



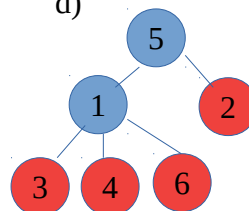
b)



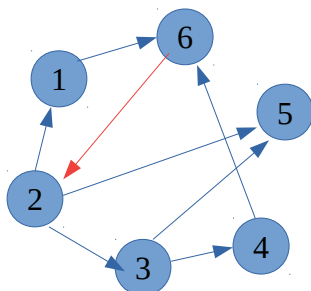
c)



d)



5)



$C1 = [2, 1, 6, 2]$
 $C2 = [2, 3, 4, 6, 2]$

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 valoarea afișată dacă se citește numărul 1527894. (6p.)

4188262

- b. Scrieți trei numere care pot fi citite astfel încât, pentru fiecare dintre acestea, în urma executării algoritmului să se afișeze 1. (6p.)

3 numere de forma $9 \cdot 10^k$, $k=0,1,2,3$.

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

- d. Scrieți în pseudocod un algoritm, echivalent cu cel dat, înlocuind adecvat structura **repetă...până când** cu o structură repetitivă cu test inițial. (6p.)

```

citește n (număr natural)
m ← 0
repetă
    c ← n%10; n ← [n/10]
    dacă c=9 atunci c ← 1
    altfel
        dacă c%2=1 atunci
            c ← c+1
        ■
    m ← m*10+c
până când n=0
scrie m

```

c)

```

citește n (număr natural)
m ← 0
repetă cat timp (n!=0) executa
    c ← n%10; n ← [n/10]
    dacă c=9 atunci c ← 1
    altfel
        dacă c%2=1 atunci
            c ← c+1
        ■
    m ← m*10+c
până când n=0
scrie m

```

d)

```

#include <iostream>
using namespace std;

```

```

int main()
{
    unsigned long c,n,m;
    cin>>n;
    m=0;
    do{
        c=n%10;n=n/10;
        if(c==9) c=1;
        else
            if(c%2==1) c=c+1;
        m=m*10+c;
    }while(n!=0);
    cout<<m;
    return 0;
}

```

2. Variabila r memorează simultan următoarele date despre fiecare dintre cele 20 de rase de porumbei dintr-o volieră: un cod, reprezentând rasa, numărul de exemplare din rasa respectivă și vârstele acestora. În volieră sunt maximum 10 exemplare din fiecare rasă. Știind că expresiile C/C++ de mai jos au ca valori numere naturale și reprezintă codul și numărul de exemplare din prima rasă, respectiv vârsta celui de al 4-lea exemplar din această rasă, scrieți definiția unei structuri cu eticheta **rasa**, care permite memorarea datelor despre o rasă de porumbei, și declarați corespunzător variabila r . (6p.)

```

r[0].cod      r[0].nrExemplare      r[0].varsta[3]

```

```

struct rasa{
    unsigned int cod, nrExemplare, varsta[11];
}r[21];

```

3. Variabilele i și j sunt de tip întreg, iar variabila a memorează un tablou bidimensional cu 5 linii și 7 coloane, numerotate începând de la 0, cu elemente numere întregi, inițial toate egale cu 0. (6p.)

Fără a utiliza alte variabile decât cele menționate, scrieți o secvență de instrucțiuni prin care se transformă în memorie tabloul, astfel încât orice element aflat pe prima linie sau pe ultima coloană să aibă valoarea 1 și oricare alt element din tablou să fie egal cu ultima cifră a sumei celor două elemente alăturate lui, aflate pe aceeași linie dar pe coloana din dreapta, respectiv pe aceeași coloană, dar pe linia anterioară.

```

1 1 1 1 1 1 1
7 6 5 4 3 2 1
8 1 5 0 6 3 1
4 6 5 0 0 4 1
0 6 0 5 5 5 1

```

```

#include <iostream>
using namespace std;
int a[5][7];
int main()
{
    int i,j;
    for(i=0;i<5;i++)
        for(j=0;j<7;j++)
            a[i][j]=0;

```

```

    for(i=0;i<7;i++) a[0][i]=1;
    for(i=0;i<5;i++) a[i][6]=1;
    for(i=1;i<5;i++)
        for(j=5;j>=0;j--)
            a[i][j]=(a[i][j+1]+a[i-1][j])%10;

```

```

    for(i=0;i<5;i++)
    {
        for(j=0;j<7;j++)
            cout<<a[i][j]<<' ';
        cout<<endl;
    }
    return 0;
}

```

1. Două numere a și b ($a < b$) sunt numite **divizori pereche** ai unui număr natural n dacă $a \cdot b = n$. Subprogramul **perechi** are un singur parametru, n , prin care primește un număr natural ($n \in [2, 10^9]$). Subprogramul afișează pe ecran toate perechile distincte de numere naturale cu paritate diferită cu proprietatea că sunt divizori pereche ai lui n . Fiecare pereche este afișată încadrată între paranteze drepte, numerele din pereche fiind afișate în ordine strict crescătoare, separate printr-un spațiu, iar dacă nu există astfel de perechi, se afișează pe ecran mesajul **nu exista**. Scrieți definiția completă a subprogramului.
- Exemplu:** dacă $n=12$ se afișează pe ecran, nu neapărat în această ordine, [1 12] [3 4] iar dacă $n=9$ se afișează pe ecran **nu exista** (10p.)

```
void perechi(unsigned long n)
{
    int exista=0;
    for(int i=1;i<n/2;i++)
        for(int j=i+1;j<=n;j++)
            if(i*j==n && i%2!=j%2 && i<j)
                {cout<< '[' << i << ' ' << j << " ] ";
                 exista=1;}
    if(!exista) cout<< "nu exista";
}
```

2. Un text are cel mult 100 de caractere și este format din cuvinte și numere, separate prin câte un spațiu. Cuvintele sunt formate numai din litere ale alfabetului englez. Toate numerele sunt reale și sunt formate numai din parte întreagă sau din parte întreagă și parte fracționară, separate prin virgulă (,), numerele negative fiind precedate de semnul minus (-). Cel puțin unul dintre numerele reale este negativ. Scrieți un program C/C++ care citește de la tastatură textul, pe care îl transformă în memorie, înlocuind fiecare număr negativ cu valoarea sa absolută. Programul afișează apoi pe ecran textul obținut.
- Exemplu:** pentru textul **Modul de -3,24 este 3,24 si modul de -15 este 15** se va afișa pe ecran textul: **Modul de 3,24 este 3,24 si modul de 15 este 15** (10p.)

```
#include <iostream>
#include <string.h>
using namespace std;

int main()
{
    char s[101];int i;
    cin.get(s,100);
    for(i=0;i<strlen(s)-1;i++)
        if(s[i]=='-' && strchr("0123456789",s[i+1])!=0)
        {
            strcpy(s+i,s+i+1);
            i--;
        }
    cout<<s;
    return 0;
}
```

3. Fișierul **bac.in** conține un șir de numere naturale distincte, din intervalul $[1, 10^9]$. Numerele din șir sunt separate prin câte un spațiu și cel puțin trei dintre ele au penultima cifră 2 și ultima cifră 0. Se cere să se afișeze pe ecran cele mai mici trei numere din șir cu proprietatea că au penultima cifră 2 și ultima cifră 0. Numerele determinate sunt afișate în ordine crescătoare, separate prin câte un spațiu. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate și al timpului de executare.
- Exemplu:** dacă fișierul conține numerele **9731 50 112 20 8 16 8520 3 2520 1520** pe ecran se vor afișa, în această ordine, numerele: **20 1520 2520**
- 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.in");
```

```

int main()
{
    unsigned long x,a,b,c;
    a=0;b=0;c=0;
    while(f>>x)
    {
        if(x%100==20)
        {
            if(a==0) a=x;
            else
                if(a!=0 && b==0)
                    {if(a>x){b=a;a=x;}
                     else b=x;}
            else
                if(a!=0 && b!=0 && c==0)
                {
                    if(x>b) c=x;
                    else
                        if(x>a) {c=b;b=x;}
                }
            else
            {
                if(x<a){c=b;b=a;a=x;}
                else
                    if(x>a && x<b){c=b;b=x;}
                    else
                        if(x>b && x<c) c=x;
            }
        }
    }
    cout<<a<<' '<<b<<' '<<c;
    f.close();
}

```

Fie $a < b < c$ valorile celor 3 variabile cerute. Citim pe rand valorile din fisier. Fie valoarea curenta x . Vom pastra in variabilele a, b, c cele mai mici valori din $\{a, b, c, x\}$. Acest lucru se va rezolva pe cazuri, prin comparatii. Vom initializa, pentru inceput, valorile lui a, b, c cu 0; aceasta initializare nu ne influenteaza rezultatul pentru ca in a, b, c vor fii pastrate valori distincte, cu ultimele doua cifre egale cu 20.

Programul contine doar o structura repetitiva, timpul de executie pentru program este mic, complexitatea acestuia este $O(n)$ – unde n reprezinta numarul de elemente din fisier.

Programul necesita putin spatiu de memorie pentru retinerea valorilor variabilelor folosite in program.

