SUBIECTUL I **(30 de puncte)**

Pentru itemul 1, scrieţi pe foaia de examen litera corespunzătoare răspunsului corect.

1. Variabilele **x**, **y** și **z** sunt de tip întreg și memorează câte un număr natural nenul. Dacă expresia C/C++ alăturată are valoarea **1**, indicați șirul crescător format cu valorile acestor variabile, în ordinea precizată mai jos. **(4p.)**

```
z<x && 2*z==3*y
```

a. x, y, z b. y, z, x c. z, x, y d. z, y, x

2. Algoritmul alăturat este reprezentat în pseudocod.

S-a notat cu **a%b** restul împărţirii numărului natural **a** la numărul natural nenul **b** şi cu **[a]** partea întreagă a numărului real **a**.

```
citeşte n
   (număr întreg)
m←0
p←1
x←0
┌dacă n<0 atunci
│ n←-n
└■
┌repetă
│ c←n%10
│ n←[n/10]
│ ┌dacă c>m atunci
│ │ m←c
│ └■
│ x←m*p+x
│ p←p*10
└până când n=0
scrie x
```

a) Scrieţi numărul afişat dacă se citeşte valoarea **10523**. **(6p.)**

55533

b) Scrieţi patru numere întregi care pot fi citite astfel încât, în urma executării algoritmului, pentru fiecare dintre acestea, să se afișeze numărul **722**. **(4p.)**

oricare patru dintre
722, -722, 712, -712, 702, -702

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

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

c)

```
citeşte n
   (număr întreg)
m←0
p←1
x←0
┌dacă n<0 atunci
│ n←-n
└■
cat timp n!=0 exeuta
│ c←n%10
│ n←[n/10]
│ ┌dacă c>m atunci
│ │ m←c
│ └■
│ x←m*p+x
│ p←p*10
└■
scrie x
```

d)

```
#include<iostream>
using namespace std;
int main()
{
    int n,m,p,x,c;
    cin>>n;
    m=0;
    p=1;
    x=0;
    if(n<0) n=-n;
    do {
      c=n%10;
      n=n/10;
      if(c>m) m=c;
      x=m*p+x;
      p=p*10;
    }while(n!=0);
cout<<x;
return 0;
}
```

SUBIECTUL al II-lea **(30 de puncte)**

Pentru fiecare dintre itemii 1 şi 2 scrieţi pe foaia de examen litera corespunzătoare răspunsului corect.

1. Expresia **strlen("2018")** are valoarea: **(4p.)**

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

2. Un graf orientat este complet dacă pentru oricare două vârfuri **i** și **j** ale sale există fie ambele arce **(i,j)** și **(j,i)**, fie doar unul dintre acestea.
Un graf orientat are **5** vârfuri și **20** de arce. Pentru a obține un graf parțial al său cu două componente tare conexe, fiecare dintre acestea fiind grafuri complete, unul cu **3** vârfuri, iar celălalt cu **2** vârfuri, numărul minim de arce care pot fi eliminate este: **(4p.)**

a. 2 **b.** 3 **c.** 6 **d.** 10

3. În declararea alăturată, câmpurile **cat** şi **rest** memorează câtul, respectiv restul împărțirii a două numere naturale nenule.

```
struct impartire
{ int cat;
  int rest;
}rezultat;
int x;
```

```
if(x!=0) {
    rezultat.cat=2018/x; rezultat.rest=2018%x;}
    else cout<<"impartire nepermisa";
```

Scrieţi o secvență de instrucțiuni în urma executării căreia se memorează în variabila **rezultat** câtul și restul împărțirii întregi a numărului **2018** la numărul natural memorat în variabila **x**, dacă acesta este nenul, sau se afișează pe ecran mesajul **impartire nepermisa**, în caz contrar. **(6p.)**

4. Un arbore cu **8** noduri, numerotate de la **1** la **8**, este reprezentat prin matricea de adiacență alăturată. Scrieţi trei noduri care pot fi alese drept rădăcină astfel încât fiecare nod să admită cel mult doi descendenți direcți (fii). **(6p.)**

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

oricare trei dintre nodurile
3, 4, 5, 7, 8

5. Scrieţi un program C/C++ care citeşte de la tastatură un număr natural **n** ($n \in [2, 10^2]$) și un șir de **n** numere naturale din intervalul $[0, 10^4]$ și construiește în memorie un tablou bidimensional cu **n** linii şi **n** coloane, numerotate începând de la **0**, astfel încât parcurgând orice coloană numerotată cu un număr par, de jos în sus, sau orice coloană numerotată cu un număr impar, de sus în jos, se obține șirul citit, ca în exemplu. Programul afișează pe ecran tabloul obținut, fiecare linie a tabloului pe câte o linie a ecranului, elementele de pe aceeași linie fiind separate prin câte un spațiu.

Exemplu: dacă **n=4**, iar șirul citit este **7 2 5 3**, se obține tabloul alăturat. **(10p.)**

3	7	3	7
5	2	5	2
2	5	2	5
7	3	7	3

```
#include <iostream>
using namespace std;
int a[101][101],b[101],n;
int main()
{
   int i,j;
        cin>>n;
        for(i=1;i<=n;i++)
            cin>>b[i];

        for(j=1;j<=n;j++)
          for(i=1;i<=n;i++)
```

```
            if(j%2==0) a[i][j]=b[i];
                  else a[n-i+1][j]=b[i];
          for(i=1;i<=n;i++)
          {
            for(j=1;j<=n;j++)
              cout<<a[i][j]<<' ';
            cout<<endl;
          }
return 0;
}
```

SUBIECTUL al III-lea **(30 de puncte)**

Pentru itemul 1, scrieţi pe foaia de examen litera corespunzătoare răspunsului corect.

1. Subprogramul **f** este definit alăturat. Indicați ce se afișează în urma apelului de mai jos.
f(7);
(4p.)

```
void f(int n)
{ cout<<n%2;  |  printf("%d",n%2);
  if(n>=3) f(n-3);
}
```

a. 10 **b.** 010 **c.** 101 **d.** 1010

Scrieţi pe foaia de examen răspunsul pentru fiecare dintre cerinţele următoare.

2. Utilizând metoda backtracking, se generează toate posibilitățile de a forma seturi de câte **5** instrumente de scris distincte din mulţimea {**stilou**, **pană**, **toc**, **creion**, **pensulă**}, astfel încât în fiecare set creionul precede stiloul și pana. Două seturi sunt distincte dacă instrumentele sunt dispuse în altă ordine.
Primele cinci soluţii generate sunt, în această ordine, (**toc**, **creion**, **stilou**, **pană**, **pensulă**), (**toc**, **creion**, **stilou**, **pensulă**, **pană**), (**toc**, **creion**, **pană**, **stilou**, **pensulă**), (**toc**, **creion**, **pană**, **pensulă**, **stilou**), (**toc**, **creion**, **pensulă**, **stilou**, **pană**). Scrieţi cea de a şasea şi cea de a şaptea soluţie, în ordinea generării acestora. **(6p.)**

(toc, creion, pensulă, pană, stilou)
(toc, pensulă, creion, stilou, pană)

3. Subprogramul **interval** are un singur parametru, **n**, prin care primeşte un număr natural (**n∈[3,10⁶]**). Subprogramul returnează cel mai mic număr natural **x** (**n<x**) care **NU** este prim, cu proprietatea că în intervalul **[n,x]** există un singur număr prim.
Scrieţi definiţia completă a subprogramului.
Exemplu: dacă **n=8**, subprogramul returnează numărul **12**. **(10p.)**

```
//vom gasi primul numar prim mai mare ca n; acesta este x-1
/*functia prim verifica daca numarul n trimis ca parametru este un numar prim */
int prim(unsigned long int n)
{
        if(n<2) return 0;
        for(int i=2;i<=n/2;i++)
          if(n%i==0) return 0;
   return 1;
}

unsigned long int interval (unsigned long int n)
{
        int i=n+1;
   while(prim(i)==0) i++;
   return i+1;
}
```

4. Primii termeni ai șirului definit alăturat (unde **n** este un număr natural nenul) sunt:
0, 3, 8, 15, 24, 35, 48, 63, 80 . . . .

$$f_n=\begin{cases}0 & \text{dacă } n=1\\ 3 & \text{dacă } n=2\\ 2\cdot f_{n-1}-f_{n-2}+2 & \text{altfel}\end{cases}$$

Se citesc de la tastatură două numere naturale din intervalul $[0,10^9]$, **x** și **y**, reprezentând valorile a doi termeni aflați pe **poziții consecutive** în şirul dat (**x<y**), şi se cere să se scrie în fişierul text **bac.txt**, în ordine strict descrescătoare, separați prin câte un spațiu, toţi termenii şirului mai mici sau egali cu **y**.
Proiectați un algoritm eficient din punctul de vedere al timpului de executare și al memoriei utilizate.
Exemplu: dacă se citesc numerele
48 63
fişierul **bac.txt** conţine numerele
63 48 35 24 15 8 3 0
a) Descrieţi în limbaj natural algoritmul proiectat, justificând eficienţa acestuia. **(2p.)**
b) Scrieţi programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

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

int main()
{
	int x,y,c;
	cin>>x>>y;
	c=2*x+2-y;
	cout<<y<<' '<<x<<' ';
	while(c>0)
	{
	cout<<c<<' ';
	y=x;
	x=c;
	c=2*x+2-y;
	}
	cout<<0;
	return 0;
}
```

Prima metoda
Presupunem ca x si y sunt ultimii doi termeni ai sirului. Deci, conform formulei date pentru n>2

f(n)=2*f(n-1)-f(n-2)+2 daca am vrea sa calculam pe f(n) atunci
f(n)=2*y-x+2.
Insa, noi avem nevoie de termenul dinainte de x. Fie acesta c.
Deci, ultimii trei termeni ar fi: y>x> c.

Pentru calculul lui f(n-1) (care este egal cu y) formula ar fi fost:
f(n-1)=2*f(n-2)-f(n-3)+2 <=> y=2*x-c+3 <=> c=y-2*x+2.

Acum avem toti termenii pentru a putea calcula termenii aflati la pasul anterior.
Algoritmul nu foloseste multe variabile iar structura repetitiva while se executa de putine ori, intr-un numar finit de pasi ceea ce duce la putina memorie ocupata pentru variabile si un timp scurt de executie.

A doua metoda (nu e optima din punct de vedere al spatiului de memorie ocupat)
Vom genera, pe rand, termenii pana ajungem la y. Vom retine acesti termeni intr-un vector apoi vom scrie termenii vetorului in ordinea inversa a lor de aparitie in sir.