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. Indicați o expresie C/C++ care are valoarea `1` dacă şi numai dacă numărul natural memorat în variabila întreagă `n` este divizibil cu `2` şi cu `5`.

a. `!(n%2==1 || n%5!=0)` **b.** `!(n/2==1 && n/5!=0)`

c. `n%2==0 || !(n%5==0)` **d.** `n/2==0 && !(n/5==0)`

2. Subprogramul `f` este definit alăturat. Indicați valoarea `f(102030)`.

```
int f (int x)
{ if(x>20) return 20+f(x/10);
  return 2020;
}
```

a. `1010` **b.** `2020` **c.** `2100` **d.** `3200`

3. Utilizând metoda backtracking, se generează toate numerele impare de cel mult trei cifre din mulţimea `{0, 1, 2, 3}`. Primele `8` soluţii generate sunt, în această ordine: `1`, `101`, `103`, `11`, `111`, `113`, `121`, `123`. Cea de a `12`-a soluţie generată este:

a. `13` **b.** `31` **c.** `133` **d.** `201`

4. Un arbore cu `10` noduri, numerotate de la `1` la `10`, este reprezentat prin vectorul de „taţi” `(2,8,2,9,8,9,0,7,7,9)`. Indicați câte dintre nodurile arborelui au exact doi fii.

a. `2` **b.** `3` **c.** `5` **d.** `6`

5. Un graf neorientat cu `20` de noduri are `100` de muchii. Numărul de muchii ce trebuie adăugate, pentru ca graful obţinut să fie complet, este:

a. `10` **b.** `50` **c.** `90` **d.** `100`

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

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

```
citeşte n (număr natural)
p←1; m←0; k←0
┌cât timp n≠0 execută
│ citeşte x (număr natural)
│┌pentru i←1,k execută
│└ x←[x/10]
│┌dacă x≠0 atunci c←x%10
│└altfel c←n%10
│ m←c*p+m
│ n←[n/10]
│ p←p*10; k←k+1
└
scrie m
```

a. Scrieţi valoarea care se afişează în urma executării algoritmului dacă se citesc, în această ordine, numerele `12345`, `780`, `921`, `4013`, `75`, `100214`. 2020 **(6p.)**

b. Dacă pentru `n` se citeşte numărul `49`, scrieți două seturi de date care pot fi citite în continuare astfel încât, pentru fiecare dintre acestea, în urma executării algoritmului, să se afişeze `49`. **(6p.)**
fie a si b cele doua numere cerute:
a=0 sau a are cifra unităților 9
- b are cifra zecilor 4 sau este format dintr-o singura cifra

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

d. Scrieţi în pseudocod un algoritm, echivalent cu cel dat, care să conțină o singură instrucțiune repetitivă. **(6p.)**

d)

```
citeşte n (număr natural)
p←1; m←0; k←0
┌cât timp n≠0 execută
│ citeşte x (număr natural)
│┌pentru i←1,k execută
│└ x←[x/10]
│┌dacă x≠0 atunci c←x%10
│└altfel c←n%10
│ m←c*p+m
│ n←[n/10]
│ p←p*10; k←k+1
└
scrie m
```

x = x/p;

c)

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

2. Variabila **t** memorează coordonatele reale (abscisa și ordonata), în planul **xOy**, ale fiecăruia dintre cele trei vârfuri **A**, **B** şi **C** ale unui triunghi. Știind că expresiile C/C++ de mai jos au ca valori abscisa vârfului **A** respectiv ordonatele vârfurilor **B** și **C** ale triunghiului, scrieți definiția unei structuri cu eticheta **triunghi**, care permite memorarea datelor precizate, și declarați corespunzător variabila **t**.

t.A.x t.B.y t.C.y **(6p.)**

```
struct Punct2D {
      int x;
      int y;
};
typedef struct Punct2D PUNCT;
```

```
typedef triunghi {
  PUNCT A;
  PUNCT B;
  PUNCT C;
}t;
```

3. În secvenţa alăturată, variabila **a** memorează un şir cu cel mult 100 de caractere, iar variabilele **i** şi **k** sunt de tip întreg.
Scrieţi ce se afişează pe ecran în urma executării secvenţei.
8viCtORIe **(6p.)**

```
k='a'-'A';
strcpy(a,"VIcToriE");
cout<<strlen(a); | printf("%d", strlen(a));
for(i=0;i<strlen(a);i++)
  if(a[i]>='A' && a[i]<='Z') a[i]=a[i]+k;
  else a[i]=a[i]-k;
cout<<a;  | printf("%s", a);
```

SUBIECTUL al III-lea (30 de puncte)

1. Subprogramul **putere** are trei parametri:

- **n**, prin care primește un număr natural din intervalul **[1,10⁹]**;
- **d** și **p**, prin care furnizează divizorul prim, **d**, care apare la cea mai mare putere, **p**, în descompunerea în factori primi a lui **n**; dacă există mai mulți astfel de divizori se afișează cel mai mare dintre ei.

Scrieți definiția completă a subprogramului.
Exemplu: dacă **n=10780**, atunci, în urma apelului, **d=7** şi **p=2** (**10780=2²·5·7²·11**). **(10p.)**

```
void putere(unsigned long n, unsigned long &d, unsigned long &p)
{
unsigned long pp,dd;
p=0,d=2;
dd=2;
while(n>1)
{
        pp=0;
        while(n%dd==0)
                {pp++;n=n/dd;}
        if(p<=pp){d=dd;p=pp;}
        dd=dd+1;
}
}
```

2. Scrieţi un program C/C++ care citeşte de la tastatură două numere naturale din intervalul **[2,20]**, **n** și **k**, şi construieşte în memorie un tablou bidimensional cu **n** linii şi **n·k** coloane, numerotate începând cu **1**, astfel încât fiecare linie **i** (**i∈[1,n]**) memorează un şir crescător de termeni cu proprietatea că primul termen este **i**, fiecare valoare apare în şir de exact **k** ori și oricare doi termeni alăturați au valori egale sau consecutive.

Programul afişează pe ecran tabloul construit, fiecare linie a tabloului pe câte o linie a ecranului, cu valorile aflate pe aceeaşi linie separate prin câte un spaţiu.
Exemplu: dacă **n=4** și **k=3**, se afişează pe ecran tabloul alăturat. **(10p.)**

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

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

int n,k,a[100][400];
int main()
{
        int i,j,p,m;
cin>>n>>k;
for(i=1;i<=n;i++)
{
        j=1;m=i;
```

```
        while(j<=n*k)
        {
                for(p=1;p<=k;p++)
                        {a[i][j]=m;j++;}
                m++;
        }
}

for(i=1;i<=n;i++)
{
        for(j=1;j<=n*k;j++)
                cout<<a[i][j]<<' ';
        cout<<endl;
}
return 0;
}
```

3. Se consideră şirul **1, 1, 2, 5, 13, 34, 89, 233, 610 . . . .**
definit astfel: $f_1=f_2=1$, $f_n=3\cdot f_{n-1}-f_{n-2}$ (unde **n** este un număr natural $n\geq 3$):
Se citesc de la tastatură două numere naturale **x** şi **y** ($x\leq y\leq 10^9$), valorile a doi termeni aflaţi pe **poziţii consecutive** în şirul dat, şi se cere să se scrie în fişierul text **bac.txt**, în ordine descrescătoare, separaţi prin câte un spaţiu, toţi termenii şirului care sunt mai mici sau egali cu **y**. Proiectaţi un algoritm eficient din punctul de vedere al memoriei utilizate şi al timpului de executare.
Exemplu: dacă se citesc numerele **89 233**
fişierul **bac.txt** conţine numerele **233 89 34 13 5 2 1 1**
a. Scrieţi programul C/C++ corespunzător algoritmului proiectat. **(8p.)**
b. Descrieţi în limbaj natural algoritmul proiectat, justificând eficienţa acestuia. **(2p.)**

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

ofstream f("bac.txt");
int main()
{
int x,y,c;
cin>>x>>y;
f<<y<<' '<<x<<' ';
c=3*x-y;
while(x!=1)
{
f<<c<<' ';
y=x;
x=c;
c=3*x-y;
}
f<<c;
f.close();
return 0;
}
```

Notam f(n-1)=x, f(n)=y si f(n-2)=c, pentru n>2;
expresia devine
y=3*x-c;
pe x si y ii cunoastem; prin urmare, putem sa-l aflam pe c (penultimul termen din sir):
c=3*x-y;
Vom parcurge termenii din sir, plecand de la cel mai mare la cel mai mic avand in vedere aceasta metoda.
Algoritmul nu consuma memorie multa deoarce se folosesc putine variabile si nu se folosesc tablouri nu numar variabil de elemente.
Timpul de executie pentru algoritm este mic deoarece nu avem instructiuni repetitive imbricate.

O alta metoda implica folosirea tabourilor unidimensionale. Vom retine intr-un tablou undimensional elementle sirului incepand cu cel mai mic pana la termenul y apoi vom afisa aceste elemente in fisier.