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 4 şi cu 5.

   a. `!(n%4!=0 || n%5!=0)` b. `!(n/4==1 || n/5!=0)`

   c. `n%4==0 && !(n%5==0)` d. `n/4==0 && !(n/5==0)`

2. Subprogramul `f` este definit alăturat. Indicați valoarea `f(200200)`.

```
int f (int x)
{ if(x>20) return 2*f(x/10);
  return 20;
}
```

   a. 160 b. 202 c. 210 d. 320

3. Utilizând metoda backtracking, se generează toate numerele impare de cel mult trei cifre din mulţimea {5, 6, 7, 8}. Primele 8 soluţii generate sunt, în această ordine: 5, 55, 555, 557, 565, 567, 57, 575. Cea de a 12-a soluţie generată este:

   a. 65 b. 67 c. 587 d. 655

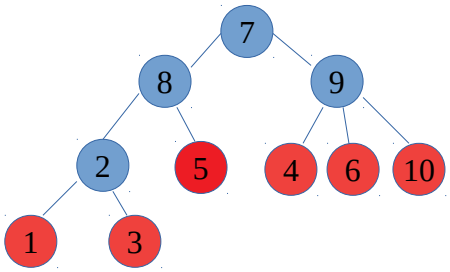
4. Un arbore cu 10 noduri, numerotate de la 1 la 10, este reprezentat prin vectorul de „taţi" (2,8,2,9,8,9,0,7,7,9). Indicați câte dintre nodurile arborelui sunt „frunze".

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

5. Un graf neorientat are 7 noduri și 20 de muchii. Indicați numărul minim de noduri care pot fi eliminate pentru ca subgraful obţinut să fie complet.

   a. 0 b. 1 c. 2 d. 3

4.



5.
Fie x numarul de noduri pentru garful complet.
Deci, 20>=x(x-1)/2
=> x=6 noduri,
graful complet cu 6 noduri are 6*5/2=15 muchii
Prin urmare, 6 din cele 7 noduri pot forma
un subgraf complet; 5 din cele 6 noduri din subgraf au o muchie incidenta celui de-al 7-lea nod.
Raspuns: 6

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)
│ i←k
│┌cât timp i≠0 execută
│└ x←[x/10]; i←i-1
│┌dacă x=0 atunci c←n%10
│└altfel c←x%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, 25, 2070, 49, 270135, 21. 10375 **(6p.)**

   b. Dacă pentru `n` se citește numărul 100, scrieți un set de date din intervalul $[10^3, 10^4)$ care pot fi citite în continuare, astfel încât, în urma executării algoritmului, să se afișeze 10. **(6p.)**

   3 numere a, b, c din intervalul [1000,10000] unde:
   a are cifra unitatilor 0
   b are cifra zecilor 1
   c are cifra miilor 0

   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.)**

c)

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

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

d)

```
citește n (număr natural)
p←1; m←0; k←0
┌cât timp n≠0 execută
| citește x (număr natural)
| i←k   x=x/p
| ┌cât timp i≠0 execută
| | x←[x/10]; i←i-1
| └■
| ┌dacă x=0 atunci c←n%10
| |altfel c←x%10
| └■
| m←c*p+m; n←[n/10]
| p←p*10; k←k+1
└■
scrie m
```

2. Variabila **c** memorează raza și coordonatele (abscisa și ordonata), în planul **xOy**, ale centrului unui cerc. Știind că expresiile C/C++ de mai jos au valori reale, reprezentând raza, respectiv coordonatele centrului cercului, scrieți definiția unei structuri cu eticheta **cerc**, care permite memorarea datelor precizate, și declarați corespunzător variabila **c**.

c.raza c.centru.x c.centru.y **(6p.)**

```
struct centrul {
  float x,y;}

struct cerc{
struct centrul centru;
float raza;}c;
```

3. Variabila **i** este de tip întreg, iar variabila **s** poate memora un șir de cel mult **20** de caractere. Scrieți ce se afișează în urma executării secvenței alăturate. **(6p.)**

```
strcpy(s,"stilou");
cout<<s+4<<endl;
for(i=0;i<4;i++)
  s[i]=s[0]+(i-1)*(1-i%2)+3*(2*i/3-1)*(i%2);
s[4]='\0';
cout<<s;
```

096	`	112	p
097	a	113	q
098	b	114	r
099	c	115	s
100	d	116	t
101	e	117	u
102	f	118	v
103	g	119	w
104	h	120	x
105	i	121	y
106	j	122	z
107	k	123	{
108	l	124	|
109	m	125	}
110	n	126	~
111	o	127	⌂

Raspuns:
ou
rosu

s=”stilou”
Scrie ou
pentru i=0
  s[0]=s[0]+(-1)*1+3*(-1)*0=’s’+(-1)=’r’
s=”rtilou”
pentru i=1
  s[1]=s[0]+0+3*(-1)*1=’r’+(-3)=’o’;
s=”roilou”
penru i=2
  s[2]=s[0]+1=’r’+1=’s’
s=”roslou”
pentru i=3
  s[3]=s[0]+3=’r’+3=’u’
s=”rosuou”
s[4]=’\0’
s=”rosu”
Scrie “rosu”

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

- **n**, prin care primește un număr natural din intervalul **[2,10⁹]**;
- **d** și **p**, prin care furnizează divizorul prim, **d**, care apare la cea mai mică putere, **p**, în descompunerea în factori primi a lui **n**; dacă există mai mulți astfel de divizori se afișează cel mai mic dintre ei.

Scrieți definiția completă a subprogramului.

Exemplu: dacă **n=10780**, atunci, în urma apelului, **d=5** şi **p=1** ($10780=2^2\cdot5\cdot7^2\cdot11$). **(10p.)**

```
void putere(unsigned long n,unsigned long &d,int &p)
{
        unsigned long dd=2; int pp;
        d=2;
        p=36000;//initializam puterea minima cu un numar mare
        while(n!=1)
        {
                pp=0;
                while(n%dd==0){n=n/dd;pp++;}
                if(pp<p && pp!=0){d=dd;p=pp;}
                dd++;
        }}
```

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·k** linii şi **n** coloane, numerotate începând cu **1**, astfel încât fiecare coloană **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 2 3 4
1 2 3 4
1 2 3 4
2 3 4 5
2 3 4 5
2 3 4 5
3 4 5 6
3 4 5 6
3 4 5 6
4 5 6 7
4 5 6 7
4 5 6 7
```

```
#include <iostream>
using namespace std;
int main()
{
int n,k,i,j,a[100][100],p;
cin>>n>>k;
p=0;
for(i=1;i<=n*k;i++)
  {
        for(j=1;j<=n;j++)
                a[i][j]=j+p;
        if(i%k==0)p++;
  }
for(i=1;i<=n*k;i++)
{
        for(j=1;j<=n;j++)
                cout<<a[i][j]<<' ';
        cout<<endl;}

return 0;}
```

3. Șirul de mai jos este definit astfel: $f_1=1$, $f_2=2$, $f_n=3\cdot f_{n-1}-2\cdot f_{n-2}$ (unde **n** este un număr natural **n≥3**).

1, **2**, **4**, **8**, **16**, **32** . . . .

Se citește de la tastatură un număr natural **x** (**x≤10⁹**), valoare a unui termen al șirului dat, şi se cere să se scrie în fişierul text **bac.txt**, în ordine descrescătoare, separați prin câte un spațiu, toţi termenii şirului care sunt mai mici sau egali cu **x**. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate şi al timpului de executare.

Exemplu: dacă se citește numărul **16**

fişierul **bac.txt** conţine numerele **16 8 4 2 1**

a. Descrieți în limbaj natural algoritmul proiectat, justificând eficiența acestuia. **(2p.)**

b. Scrieți programul C/C++ corespunzător algoritmului proiectat. **(8p.)**

Sirul este format din sirul de numere cu puteri ale lui 2. Acest lucru e greu de observat din definitia sirului; dar din exemplul dat pentru termenii sirului si facand cateva calcule pentru verificarea acestei ipoteze putem trage concluzia ca termenii sirului sunt puteri ale lui 2 in ordine crescatoare.
Fara aceasta observatie nu putem rezolva problema, bazandu-ne doar pe definitia elementelor sirului in care nu cunoastem aceast mod de definire pentru generea sirului de puteri ale lui 2.
Observatie: elementul al n-lea din sir coincide cu $2^{(n-1)}$

Prin urmare, fie y penultimul termen al sirului f(n-1) si x ultimul termen al sirului f(n). Atunci, sirul se poate defini astfel:
f(n)=1 pentru n=1
f(n)=2 pentru n=2
f(n)=2*f(n-1) pentur n>=3 <=> x=2*y <=> y=x/2

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
unsigned x,y;
ofstream f("bac.txt");
cin>>x;
f<<x<<' ';
while(x>=2)
{
	y=x/2;
	f<<y<<' ';
	x=y;
}
f.close();
return 0;
}
```

Programul foloseste doar doua variabile pentru a afisa in ordine descrescatoare termenii sirului: x- termenul curent, y- termenul precedent; pentru trecerea la urmatoarea iteratie se reinitializeaza cei doi termeni: x devine termenul precedent de la pasul anterior iar y se calculeaza conform formulei gasite; prin urmare nu rezerva multa memorie pentru retinerea variabilelor la executia programului.
Programul are o complexitate liniara O(n); foloseste doar o structura repetitiva, deci, timul de executie este mic.