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. Expresia C/C++
`(x>=16) && !( x<17 || x>19) && (x<=20)`
are valoarea 1 dacă și numai dacă valoarea memorată de variabila întreagă **x** aparține intervalului:

a. `[16,18]` **b.** `[17,19]` c. `[18,20]` d. `[19,20]`

2. Utilizând metoda backtracking se generează toate posibilitățile de a așeza în compartimentele unei voliere porumbei de rase din mulțimea {**creți**, **iacobini**, **jucători**, **rotați**, **toboșari**}. Două soluții sunt diferite dacă ordinea raselor diferă. Primele patru soluții obținute sunt, în această ordine: (**creți**, **iacobini**, **jucători**, **rotați**, **toboșari**), (**creți**, **iacobini**, **jucători**, **toboșari**, **rotați**), (**creți**, **iacobini**, **rotați**, **jucători**, **toboșari**), (**creți**, **iacobini**, **rotați**, **toboșari**, **jucători**). Indicați penultima soluție generată.

a. (**toboșari**, **rotați**, **creți**, **iacobini**, **jucători**)

b. (**toboșari**, **rotați**, **creți**, **jucători**, **iacobini**)

c. (**toboșari**, **rotați**, **jucători**, **creți**, **iacobini**)

d. (**toboșari**, **rotați**, **jucători**, **iacobini**, **creți**)

3. Fiecare dintre variabilele **A** și **B**, declarate alăturat, memorează coordonatele pozitive (**x** abscisa, iar **y** ordonata) ale câte unui punct în sistemul de coordonate xOy, extremități ale unui segment. Indicați o expresie C/C++ care are valoarea 1 dacă și numai dacă cel puțin una dintre extremitățile segmentului precizat este în originea sistemului de coordonate xOy.

```
struct punct
{ int x,y;
} A,B;
```

a. `(A.x+A.y)*(B.x+B.y)==0` b. `(A(x)+A(y))*(B(x)+B(y))==0`

c. `(x.A+y.A)*(x.B+y.B)==0` d. `punct.A(x+y)*punct.B(x+y)==0`

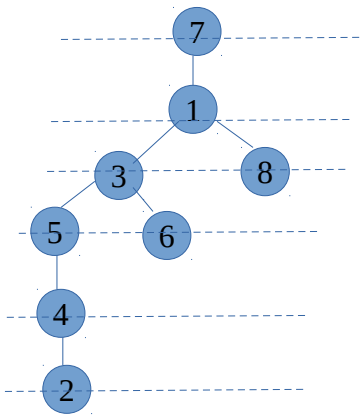
4. Într-un arbore cu rădăcină un nod se află pe nivelul **x** dacă lanțul elementar care are o extremitate în nodul respectiv și cealaltă extremitate în rădăcina arborelui are lungimea **x**. Pe nivelul 0 se află un singur nod (rădăcina).
Un arbore cu rădăcină are 8 noduri, numerotate de la 1 la 8, și muchiile `[1,3]`, `[1,7]`, `[1,8]`, `[2,4]`, `[3,5]`, `[3,6]`, `[4,5]`. Știind că rădăcina arborelui este nodul numerotat cu 7, indicați numărul de niveluri ale arborelui dat.

a. 3 b. 4 **c.** 6 d. 7

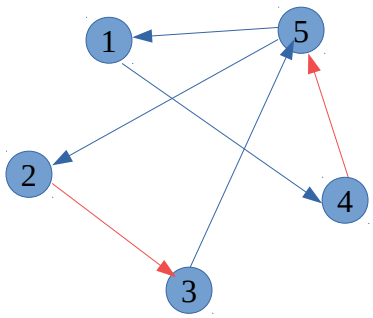
5. Un graf orientat cu 5 vârfuri, numerotate de la 1 la 5, are arcele `(1,4)`, `(3,5)`, `(5,1)`, `(5,2)`. Indicați numărul minim de arce care trebuie adăugate acestuia, astfel încât graful obținut să fie tare conex.

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

4.



5.



Un graf orientat G=(V,E) este tare conex dacă pentru orice pereche de noduri distincte (x,y) există cel puțin un drum de la x la y și există cel puțin un drum de la y la x.

SUBIECTUL al II-lea (40 de puncte)

1. **Algoritmul alăturat este reprezentat în pseudocod.**

```
citește n (număr natural)
 nr←0
┌pentru i←n,1,-1 execută
│ x←0; y←1
│ ┌pentru j←1,i execută
│ │ r←2*x-y; x←y; y←r
│ └■
│ ┌dacă y>0 atunci
│ │ nr←nr+1
│ └■
└■
scrie nr
```

a. Scrieți numărul afișat în urma executării algoritmului dacă pentru `n` se citește valoarea `5`. **2** **(6p.)**

b. Scrieți două numere din intervalul `[10,10²)` care pot fi citite astfel încât, pentru fiecare dintre acestea, în urma executării algoritmului, să se afișeze `14`. **28 29** **(6p.)**

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

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

c)

```
#include<iostream>
using namespace std;
int main()
{
        int n,i,x,y,r,j; int nr;
        cin>>n;
        nr=0;
for(i=n;i>=1;i--)
{
 x=0;y=1;
 for(j=1;j<=i;j++)
  {r=2*x-y;x=y;y=r;}
 if(y>0) nr++;
 }
cout<<nr;
return 0;
}
```

d)

```
citește n (număr natural)
nr←0
i=n;
cat timp (i>=1) executa
   x←0; y←1
   ┌pentru j←1,i execută
   │ r←2*x-y; x←y; y←r
   └■
   ┌dacă y>0 atunci
   │ nr←nr+1
   └■
   i=i-1
scrie nr
```

2. Subprogramul `f` este definit alăturat. Scrieți două numere naturale din intervalul `[1,10]`, care pot fi memorate în variabilele întregi `x1`, respectiv `x2`, astfel încât valoarea lui `f(10,x1)` să fie `5`, iar valoarea lui `f(x2,10)` să fie `1`. **(6p.)**

```
int f(int x, int y)
{ if(x>y) return x%y+f(x-y,y);
  if(x<y) return y%x+f(x,y-x);
  return 1;
}
```

x1=4
x2 = o valoare din multimea {1, 2, 5, 10}

3. Variabilele `i` și `j` sunt de tip întreg, iar variabila `a` memorează un tablou bidimensional cu `4` linii și `5` coloane, numerotate începând de la `0`, cu elemente numere întregi, inițial toate nule. Fără a utiliza alte variabile decât cele menționate, scrieți o secvență de instrucțiuni astfel încât, în urma executării acesteia, variabila `a` să memoreze tabloul alăturat. **(6p.)**

```
1  5  9 13 17
2  6 10 14 18
3  7 11 15 19
4  8 12 16 20
```

```
for(i=0;i<=3;i++)
    for(j=0;j<=4;j++)
        if(j==0) a[i][j]=i+1;
          else a[i][j]=a[i][j-1]+4;
```

SUBIECTUL al III-lea (30 de puncte)

1. Subprogramul **`divPrimMax`** are doi parametri:

- `n`, prin care primește un număr natural (`n∈[2,10⁹]`);
- `p`, prin care furnizează cel mai mare divizor prim al lui `n`.

Scrieți definiția completă a subprogramului.
Exemplu: dacă `n=2000`, în urma apelului `p=5`, deoarece $2000=2^4 \cdot 5^3$. **(10p.)**

```
void divPrimMax(unsigned long n, unsigned long &p)
{
        unsigned long d=2,k;
        p=0;
        while(n!=1)
        {
        k=0;
        while(n%d==0) {k++;n=n/d;}
        if(k!=0) p=d;
                d++;
                }
}
```

2. Într-un text cu cel mult **100** de caractere, cuvintele sunt formate din litere mici ale alfabetului englez și sunt separate prin câte un spațiu. Scrieți un program C/C++ care citește de la tastatură un text de tipul menționat și afișează pe ecran numărul de cuvinte ale sale formate dintr-un număr egal de vocale și consoane. Se consideră vocale literele din mulțimea **a**, **e**, **i**, **o**, **u**.

Exemplu: pentru textul
cuvantul consoane are un numar de patru vocale si patru consoane
se afișează pe ecran **6**. **(10p.)**

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

int main()
{
  char str[101];
  char *p;
  int i,n,nr=0,c,j,t;
  char v[50][20];
  cin.get(str,100);
  n=0;
  p = strtok(str, " ");
  while( p != NULL )
  {
    n++; strcpy(v[n],p);
    p = strtok(NULL, " ");
  }
for(i=1;i<=n;i++)
        {
                //cout<<v[i]<<' ';
                c=0; t=strlen(v[i]);
                for(j=0;j<t;j++)
                        if(strchr("aeiou",v[i][j])) c++;
                if(t==2*c) nr++; //daca in sir avem numarul de consoane egal cu cel de vocale
  }
cout<<nr;
return 0;
}
```

3. Se citesc de la tastatură două numere naturale din intervalul **[1,81]**, **p1** și **p2**, și se cere scrierea în fișierul **bac.out** a tuturor numerelor naturale cu exact **7** cifre, pentru care produsul primelor două cifre este egal cu **p1**, cele trei cifre din mijloc sunt egale între ele, iar produsul ultimelor două cifre este egal cu **p2**. Numerele apar în fișier în ordine strict crescătoare, fiecare pe câte o linie. Proiectați un algoritm eficient din punctul de vedere al memoriei utilizate și al timpului de executare.

Exemplu: dacă **p1=12**, iar **p2=8**, atunci **2633324** și **3400018** sunt două dintre cele **160** de numere cu proprietatea cerută ($2 \cdot 6=3 \cdot 4=12$ și $2 \cdot 4=1 \cdot 8=8$).

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;

ofstream f("bac.out");

int main()
{
unsigned long c1, c2, c3,i,j,k,p1,p2,s,s1;
cin>>p1>>p2;
for(i=9;i>=1;i--)
{
        if(p1%i==0 && p1/i<=9)
```

```
        {
                c1=i*10+(p1/i);
                for(j=0;j<=9;j++)
              {
               c2=j*100+j*10+j;
                    s1=c1*1000+c2;
                    for(k=9;k>=1;k--)
                      {
                          if(p2%k==0 && p2/k<=9) {
                                        c3=k*10+(p2/k);
                                        s=s1*100+c3;
                                        f<<s<<endl;
                                  // cout<<s<<' ';
                                }
                       }
               }
    }
}
return 0;
}
```

p1 si p2 sunt numere de cate doua cifre si, in acelasi timp, sunt produse de cate doua cifre.
Vom construi numarul cerut astfel:
p1- ca produs de doua cifre
p2 – ca produs de doua cifre
numarul de la mijloc – ca un numar format din aceasi cifra
Pentru fiecare dintre cele 3 subparti de numere avem nevoie doar de o cifra, cealalta/celelalte numere obtinandu-se din aceasta cifra fie prin impartirea numerelor p1 si p2 la cifra curenta fie prin construirea unui numar cu 3 cifre identice.
Dupa ce am obtinut cele 3 subparti ale numarului cu 7 cifre alipim cele 3 subparti pentru a obtine un numar cu 7 cifre.
Vom folosi toate combinatiile de cate 3 cifre posibile, cate o cifra pentru fiecare parte a numarului.
Algoritmul foloseste putine variabile, prin urmare, e nevoie de putin spatiu de memorie pentru a le retine.
Algoritmul foloseste trei instructiuni for imbricate dar valorile din for sunt pe interval scurt, de constante: 0...9. Prin urmare, timpul de executie pentru program este mic.