Temă

Teza 6

1. b) (atât diferența, cât și suma două numere cu aceeași paritate este întotdeauna un număr par)

2.

În graful G vor exista următoarele muchii

$1 \rightarrow \{2,3\}$ (muchiile $(1,2)$ și $(1,3)$)

$2 \rightarrow \{4,5\}$

$3 \rightarrow \{6,7\}$

$\vdots$

$1009 \rightarrow \{2018, 2019\}$

Fiecare dintre primele 1009 noduri este adiacent cu 2 noduri => numărul de muchii din graf este egal cu

$2 \cdot 1009 = 2018$

b)

3. Se observă că numărul total de dale este egal cu 8, iar o configurație a dalelor este considerată distinctă față de o altă configurație dacă și numai dacă există cel puțin o poziție în care dalele din cele două configurații au culori diferite.

În câte moduri se pot plasa cele 5 dale albe în configurație?

În C_8^5 moduri, deoarece dalele albe pot fi plasate în oricare 5 poziții distincte din cele 8, iar ordinea în care acestea fi plasate nu contează, deoarece culoarea șilelor introduce diferențe dintre configurații, nu poziția.

Celelalte trei șile rămase (negre) pot fi plasate, după același raționament, în C_3^3 moduri, deoarece după plasarea celor 5 șile albe au mai rămas 3 poziții libere, pe care șilele negre pot fi așezate într-un singur mod (nu contează ordinea acestora)

Răspunsul este $C_8^5 \cdot C_3^3 = 56$ de moduri

d)

4.

Numărul de muchii ale unui graf neorientat complet cu n noduri este egal cu $\frac{n(n-1)}{2}$

Pentru ca numărul de muchii crește polinomial în raport cu numărul de noduri din graf / componenta conexă, rezultă că cel mai mare număr de muchii s-ar obține pentru valoarea maximă a numărului de noduri din graf / componenta conexă.

În cazul de față, numărul maxim de noduri din care poate fi constituită o componentă conexă este egal cu 10, deoarece trebuie ca graful să aibă exact 3 componente conexe.

Rezultă că numărul maxim de muchii care pot exista în graf este egal cu $\frac{10 \cdot 9}{2} = 45$ (celelalte două componente conexe nu au muchii pentru ca sunt alcătuite din câte un nod izolat)

(5)

În urma apelului f(8) se va afișa 124123121

f(8) → f(6) → f(4) → f(2) → f(0)

f(8)	f(6)	f(4)	f(2)	f(0)
afișare 1,2,4	afișare 1,2,3	afișare 1,2	afișare 1	
f(6)	f(4)	f(2)	f(0)	

b)

Subiectul III

1.

Algoritmul descris în pseudocod determină numărul de divizori proprii ai numărului x și afișează acest număr, în cazul în care este nenul, ori 1 în cazul în care este nul (caz în care numărul x este prim sau este egal cu 1)

a) Valoarea afișată pentru 45 este 4 (45 are 4 divizori proprii: 3, 5, 9, 15)

b) Valorile din intervalul [1,9] pentru care se afișează valoarea 1 sunt:

1, 2, 3, 5, 7,4,9

c)

```
#include <iostream>
using namespace std;
int main()
{
    int x, y=0;
    cin>>x;
    for(int i=2; i<=x/2; i++)
        if(x%i==0) y++;
}
```

```
if (y==0)
  cout<<1;
else
  cout<<y;
return 0;
}
```

d) citește x
(x număr întreg)
y ← 0
i ← 2

```
┌- cât timp i ≤ x / 2 execută
│  ┌- dacă x % i = 0 atunci
│  └□ y ← y+1
│  i ← i+1
└□
┌- dacă y=0 atunci
│   scrie 1
│  altfel
│   scrie y
└□
```

2. Panta unei drepte pe care se află un segment AB, $A(x_A, y_A)$, $B(x_B, y_B)$ este egală cu $\frac{y_A - y_B}{x_A - x_B}$

Pentru ca segmentul AB să se afle pe prima bisectoare (dreapta de ecuație $y=x$), dreapta suport a segmentului AB trebuie să coincidă cu dreapta de ecuație $y=x$, deci să aibă panta egală cu 1.

```
if ((A.y - B.y)/(A.x - B.x) == 1)
```

3) Se observă relația $a_{ij} = i + 5(j-1)$

```
a[i][j]= i+ 5*(j-1);
```

Subiectul III

1)

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char str[101];
    int max_l=0, aparitii=0;
    cin.get(str,100);
    char *p = strtok(str," ");
    while(p != NULL)
    {
        if(strlen(p) > max_l || max_l==0)
        {
            max_l = strlen(p);
            aparitii = 1;
        }
        else if(strlen(p)==max_l)
            aparitii++;
        p = strtok(NULL," ");
    }
    cout << max_l << ' ' << aparitii;
    return 0;
}
```

④

```
int divizor( int n, int a[100])
{
      int cmmdc = 1;

      for( int i=0; i<n; i++)
      {
            if( i==0)
            { cmmdc= a[i];
              continue;
            }

       int x= cmmdc;
       int y= a[i];

       while( x !=y)
       {
            if( x>y)
               x = x-y;
            else
               y = y-x;
       }

       cmmdc= a;
      }
   return cmmdc;
}
```

⑤

a) Algoritmul implementat este eficient din punctul de vedere al timpului de executare având complexitatea de timp liniară, iar complexitatea de memorie este una constantă, bazându-se pe o proprietate matematică a numerelor p-compuse.

Un număr p compus se poate scrie sub forma $x = \sum_{K=0}^{p} (a+k)$, unde

a reprezintă primul număr din sumă.

Se observă că pentru ca un număr x să fie p compus, trebuie să existe o sumă $\sum_{k=0}^{p}(a+k)$ egală cu x, care să aibă proprietatea că $a \in \mathbb{N}$

$$x = \sum_{k=0}^{p} (a+k) \Leftrightarrow x = a + (a+1) + (a+2) + \dots + (a+(p-1)) \Leftrightarrow$$

$$x = pa + \frac{p(p-1)}{2} \Leftrightarrow$$

$$a = \frac{x - \frac{p(p-1)}{2}}{p}$$

Cum a trebuie să fie natural, rezultă că numărul x este p- compus dacă și numai dacă $x - \frac{p(p-1)}{2}$ este divizibil cu p. (1)

În plus, $a = \frac{x - \frac{p(p-1)}{2}}{p} \Leftrightarrow a = \frac{x}{p} - \frac{\cancel{p}(p-1)}{2\cancel{p}} \Leftrightarrow$

$a = \frac{x}{p} - \frac{p-1}{2}$, deci condiția (1) este echivalentă

cu „numărul x trebuie să fie divizibil cu p, iar p să fie impar".

```
# include <iostream>
# include <fstream>
using namespace std;
ifstream fin("BAC.TXT");
int main()
{
    int p, x;
    fin>>p;
    while( fin>>x)
    {
        if( x%p==0 && p%2==1)
          cout<< (x/p) + (p-1)/2 ;
        else
          cout<<"NU";
    }
    return 0;
}
```