МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ

“ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

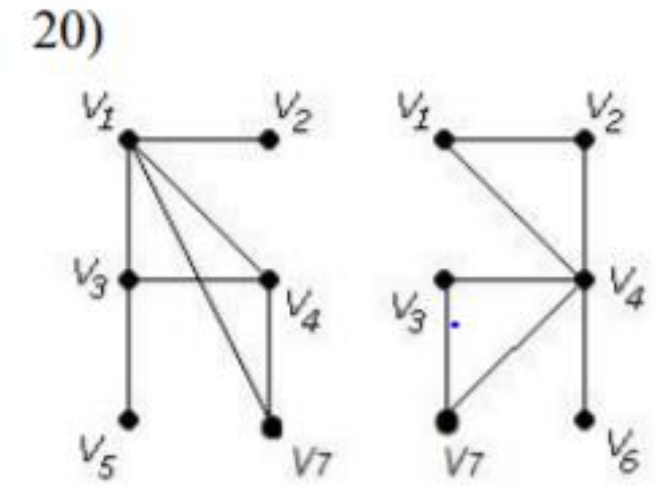
Виконав:

Студент групи: КН-113

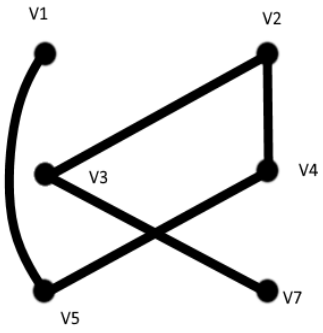
Омелюх Р.Т.

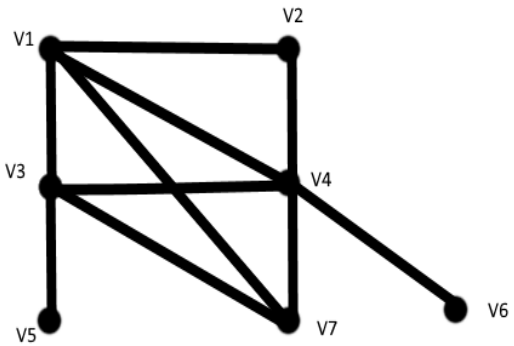
Перевірила:
Мельникова Н.І

1)Виконати наступні операції над графами: 1) знайти доповнення до першого графу, 2) об'єднання графів, 3) кільцеву сумму G1 та G2 (G1+G2), 4) розмножити вершину у другому графі, 5) виділити підграф А - що скадається з 3-х вершин в G1 6) добуток графів.

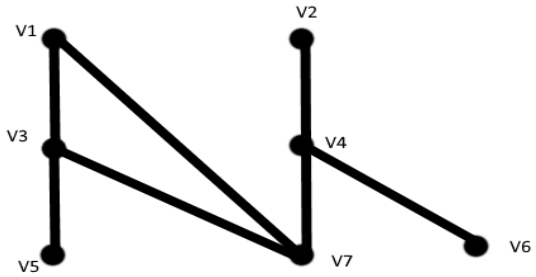


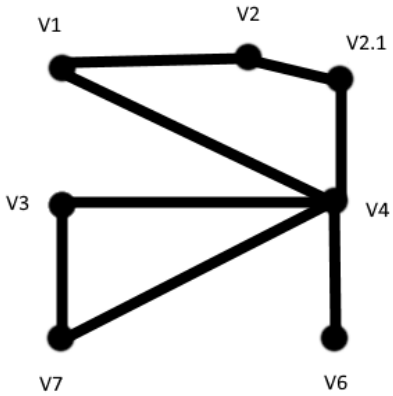
1) 2)

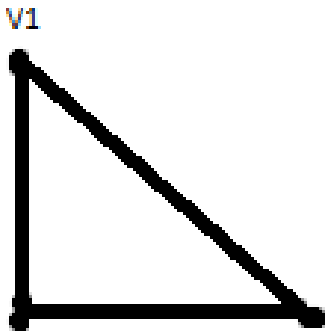




3) **4)**

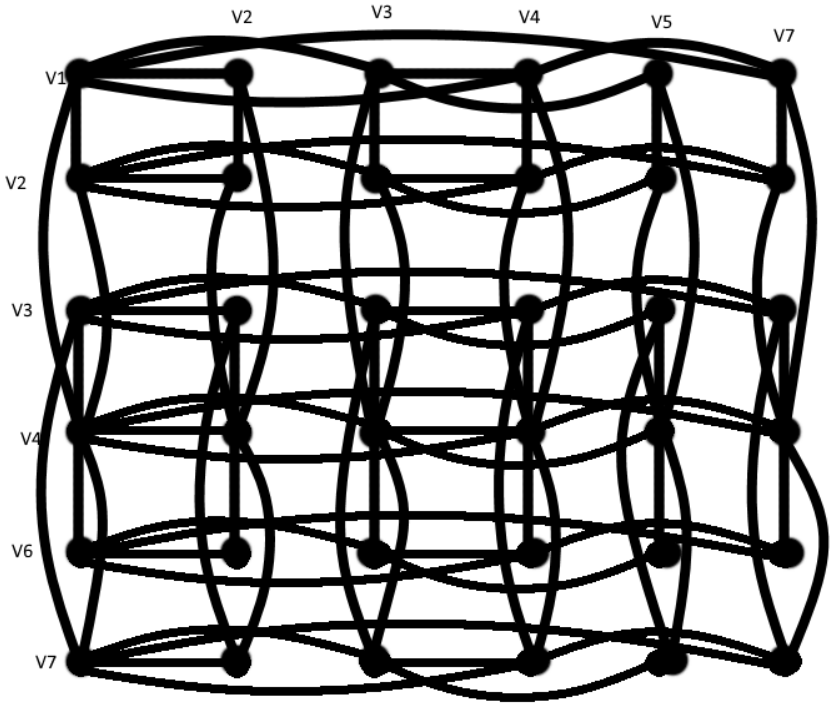




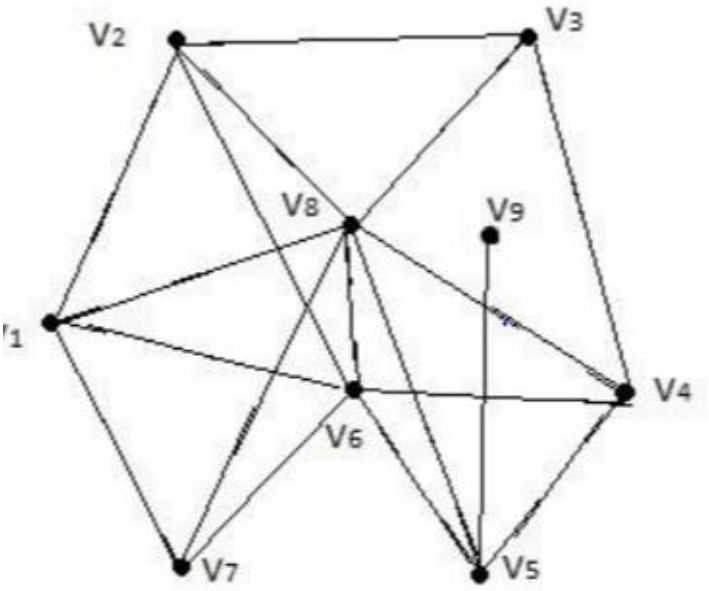


5)

6)



2.Скласти таблицю суміжності для орграфа.



	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	0	0	0	1	1	1	0
V2	1	0	1	0	0	1	0	1	0
V3	0	1	0	1	0	0	0	1	0
V4	0	0	1	0	1	1	0	1	0
V5	0	0	0	1	0	1	0	1	1
V6	1	1	0	1	1	0	1	1	0
V7	1	0	0	0	0	1	0	1	0
V8	1	1	1	1	1	1	1	0	0
V9	0	0	0	0	1	0	0	0	0

Завдання № 3

Для графа з другого завдання знайти діаметр

.D=3(V1-V8-V5-V9)

Завдання № 4

Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число) або вшир

4

V1	1	V1
V2	2	V1,v2
V8	3	V1,V2,V8
V6	4	V1,V2,V8,V6
V7	5	V1,V2,V8,V6,V7
-	-	V2,V8,V6,V7
V3	6	V2,V8,V6,V7,V3
-	-	V8,V6,V7,V3
V4	7	V8,V6,V7,V3,V4
V5	8	V8,V6,V7,V3,V4,V5
-	-	V6,V7,V3,V4,V5
-	-	V7,V3,V4,V5
	-	V3,V4,V5
	-	V4,V5
-	-	V5
V9	9	V5,V9
-	-	V9
-	-	~~0~~

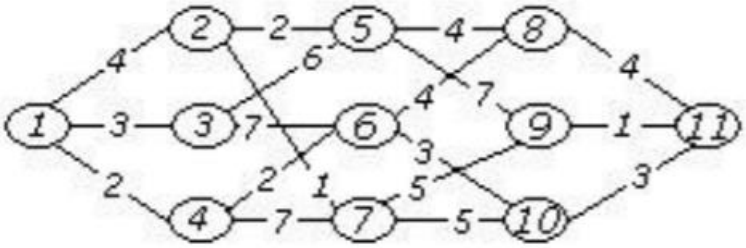
Код реалізації

```cpp
#include <iostream>
#include<bits/stdc++.h>
#include <queue> // черга
using namespace std;
int main()
{
    setlocale(LC_ALL,"Ukrainian");
  queue <int> Queue;
  int mas[9][9] = {  // матриця суміжності
  {0,1,0,0,0,1,1,1,0},
  {1,0,1,0,0,1,0,1,0},
  {0,1,0,1,0,0,0,1,0},
  {0,0,1,0,1,1,0,1,0},
  {0,0,0,1,0,1,0,1,1},
  {1,1,0,1,1,0,1,1,0},
  {1,0,0,0,0,1,0,1,0},
  {1,1,1,1,1,1,1,0,0},
  {0,0,0,0,1,0,0,0,0}, };
  int nodes[9]; // вершини графа
  for (int i = 0; i < 9; i++)
    nodes[i] = 0; //  всі вершини рівні 0
  Queue.push(0); // ставлю в чергу першу вершину
   cout<<"Черга додавання вершин"<<endl;
  while (!Queue.empty())//поки черга не пуста
  {
    int node = Queue.front(); // витягую вершину
    Queue.pop();
    nodes[node] = 2; // помічаю її як пройдену

    for (int j = 0; j < 9; j++)
    { // перевіряю для неї всі суміжні вершини
      if (mas[node][j] == 1 && nodes[j] == 0)
      { // якщо вершина суміжна і ще не в черзі
        Queue.push(j); // добавляємо її в чергу
        nodes[j] = 1; // помічаю вершину як пройдену
      }
    }
    cout<<"V"<< node + 1 << endl; // виводжу номер вершини
  }
  cin.get();
  return 0;
}
```

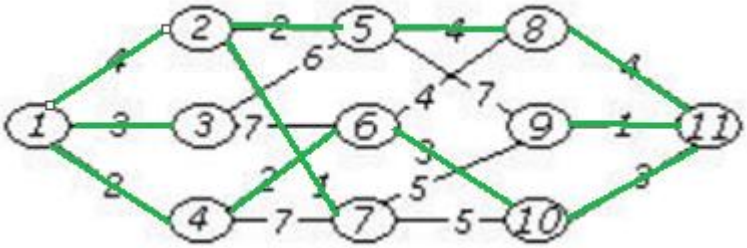
```
"E:\яЁюуЁрьрў\яю°єъ т°шЁ\bin\Debug\яю°єъ т°шЁ.exe"
Черга додавання вершин
V1
V2
V6
V7
V8
V3
V4
V5
V9
2

Process returned 0 (0x0)   execution time : 19.845 s
Press any key to continue.
```

Завдання № 5

Знайти двома методами (Краскала і Прима) мінімальне остове дерево граф



Краскала:



V(t) = {2,7,9,11,1,4,5,6,3,10,8};
E(t) = {(2,7),(9,11),(1,4),(2,5),(4,6),(1,3),(6,10),(10,11),(5,8),(1,2)};

Код реалізації:

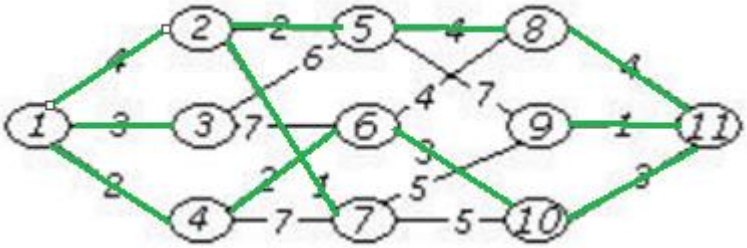
```cpp
#include <iostream>
#include <stdio.h>
using namespace std;
struct rebro {

 int leng;
 int v1;
 int v2;
 bool in = false;
};
struct mas {

 int arr[11] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
 int c = 0;
};
int in(rebro *reb, int n) {
 setlocale(LC_ALL, "Ukrainian");
 for (int i = 0; i < n; i++)
 {
  cout << "Введіть довжину " << i + 1 << " ребра: ";
  cin >> reb[i].leng;
  cout << "Введіть першу суміжну вершину з " << i + 1 << " ребром: ";
  cin >> reb[i].v1;
  cout << "Введіть другу суміжну вершину з " << i + 1 << " ребром: ";
  cin >> reb[i].v2;
  cout << endl;
 }
 return 0;
}
int main() {

 setlocale(LC_ALL, "Ukrainian");
 setlocale(LC_ALL, "Ukrainian");
 int n = 0, x = 100, y = 100;
 cout << "Введіть кількість ребер у графі: ";
 cin >> n;
 int z;
 cout << "Введіть кількість вершин у графі: ";
 cin >> z;
cout << endl;
 rebro *reb = new rebro[n];
 mas inn[5];
 in(reb, n);
 for (int i = 0; i < n - 1; i++)
 {
  for (int j = 0; j < n - 1; j++)
  {
   if (reb[j].leng > reb[j + 1].leng) { swap(reb[j].leng, reb[j + 1].leng); swap(reb[j].v1, reb[j + 1].v1); swap(reb[j].v2, reb[j + 1].v2); }
  }
 }
 int c = -1;
 for (int i = 0; i < n; i++)
 {
  for (int j = 0; j < 5; j++)
  {
   for (int k = 0; k < z; k++)
   {
    if (reb[i].v1 == inn[j].arr[k]) { x = j; goto point0;; }
   }
  }
point0:;
  for (int j = 0; j < 5; j++)
  {
   for (int k = 0; k <z; k++)
```

```
63      for (int k = 0; k <z; k++)
64      {
65       if (reb[i].v2 == inn[j].arr[k]) { y = j; goto point1; }
66      }
67     }
68    point1:;
69     if (x != y && x == 100) { inn[y].arr[inn[y].c] = reb[i].v1; inn[y].c++; }
70                                       mas main::inn
71     if (x != y && y == 100) { inn[x].arr[inn[x].c] = reb[i].v2; inn[x].c++; }
72
73     if (x != y && x != 100 && y != 100) {
74      if (x < y) {
75       for (int l = 0; l < inn[y].c; l++)
76       {
77        inn[x].arr[inn[x].c+l] = inn[y].arr[l];
78        inn[y].arr[l] = 0;
79       }
80       inn[x].c += inn[y].c;
81       inn[y].c = 0;
82      }
83
84      if (y < x) {
85       for (int l = 0; l < inn[x].c; l++)
86       {
87        inn[y].arr[inn[y].c+l] = inn[x].arr[l];
88        inn[x].arr[l] = 0;
89       }
90       inn[y].c += inn[x].c;
91       inn[x].c = 0;
92      }
93     }
94     if (x == 100 && y == 100) { c++;  inn[c].arr[inn[c].c] = reb[i].v1; inn[c].arr[inn[c].c + 1] = reb[i].v2; inn[c].c += 2; }
95      reb[i].in = true;
96      if (x == y && x != 100) { reb[i].in = false; }
97      x = 100; y = 100;
98     }
99     cout << "Щоб побудувати остове дерево мінімальної ваги, ми повинні включити в нього такі ребра: " << endl;
100    int s = 0;
101    for (int i = 0; i < n; i++)
102    {  if (reb[i].in == true) { cout << "Ребро" << ", що сполучає вершини " << reb[i].v1 << " " << reb[i].v2 << endl; s += reb[i].leng; }
103    }
104    cout << "Остове дерево мінімальної ваги для даного графа: " << s;
105    return 0;
106   }
```

Result:

```
Введіть першу суміжну вершину з 15 ребром: 7
Введіть другу суміжну вершину з 15 ребром: 10

Введіть довжину 16 ребра: 4
Введіть першу суміжну вершину з 16 ребром: 8
Введіть другу суміжну вершину з 16 ребром: 11

Введіть довжину 17 ребра: 1
Введіть першу суміжну вершину з 17 ребром: 9
Введіть другу суміжну вершину з 17 ребром: 11

Введіть довжину 18 ребра: 3
Введіть першу суміжну вершину з 18 ребром: 10
Введіть другу суміжну вершину з 18 ребром: 11

Щоб побудувати остове дерево мінімальної ваги, ми повинні включити в нього такі ребра:
Ребро, що сполучає вершини 2 7
Ребро, що сполучає вершини 9 11
Ребро, що сполучає вершини 1 4
Ребро, що сполучає вершини 2 5
Ребро, що сполучає вершини 4 6
Ребро, що сполучає вершини 1 3
Ребро, що сполучає вершини 6 10
Ребро, що сполучає вершини 10 11
Ребро, що сполучає вершини 1 2
Ребро, що сполучає вершини 5 8
Остове дерево мінімальної ваги для даного графа: 25
Process returned 0 (0x0)   execution time : 177.408 s
Press any key to continue.
```

Прима

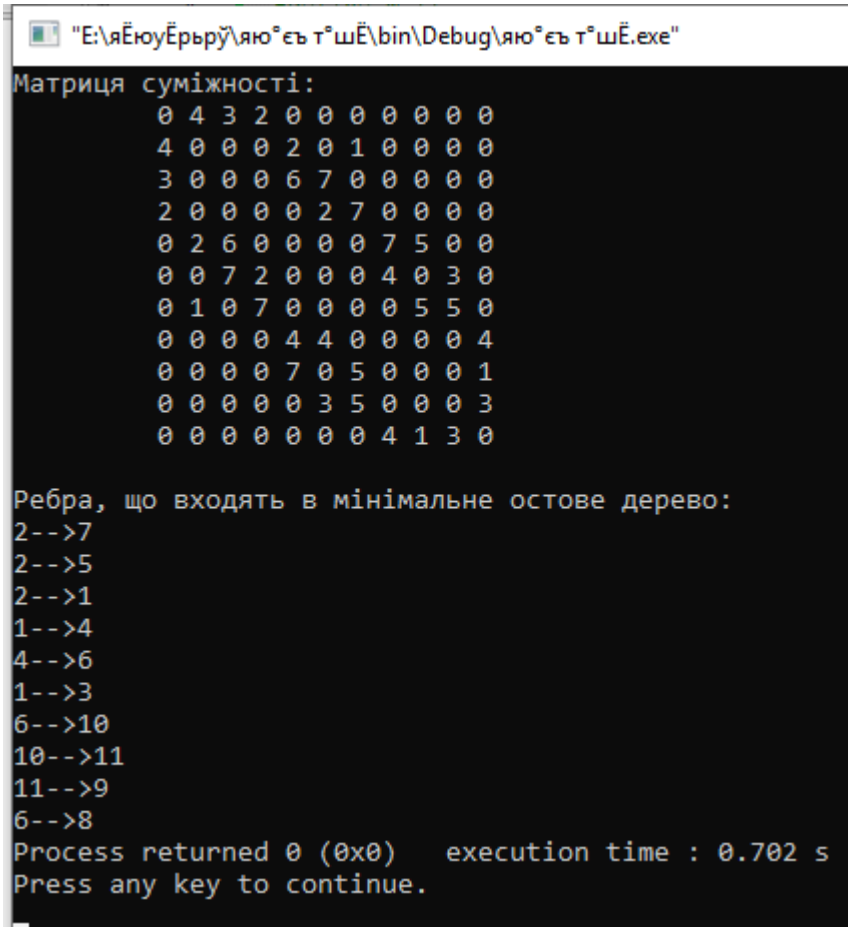



```
V(t) = {2,7,5,1,4,6,3,10,11,9,8};
E(t) = {(2,7),(2,5),(2,1),(1,4),(4,6),(1,3),(6,10),(10,11),(11.9),(11,8)};
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#define m 11
int main()
{
setlocale(LC_CTYPE,"ukr");
int graph[m][m]={
   {0,4,3,2,0,0,0,0,0,0,0},
   {4,0,0,0,2,0,1,0,0,0,0},
   {3,0,0,0,6,7,0,0,0,0,0},
   {2,0,0,0,0,2,7,0,0,0,0},
   {0,2,6,0,0,0,0,7,5,0,0},
   {0,0,7,2,0,0,0,4,0,3,0},
   {0,1,0,7,0,0,0,0,5,5,0},
   {0,0,0,0,4,4,0,0,0,0,4},
   {0,0,0,0,7,0,5,0,0,0,1},
   {0,0,0,0,0,3,5,0,0,0,3},
   {0,0,0,0,0,0,0,4,1,3,0}
};
   int i,j,p=0,q=0;
   printf("Матриця суміжності:\n\t ");
   for(i=0;i<m;i++){
      for(j=0;j<m;j++)
            printf("%d ",graph[i][j]);
   printf("\n\t ");}
   printf("\nРебра, що входять в мінімальне остове дерево:\n");
   int visit[m]={0};
   int arr[m]={1,2,3,4,5,6,7,8,9,10,11};
```

```
    int min;
    int flag=0;
    for(i=0;i<m;i++){
        for(j=0;j<m;j++){
            if(flag==0&&graph[i][j]!=0){
                flag=1;
        p=i;
        q=j;
        min=graph[p][q];
    }
    else if(flag==1&&graph[i][j]<min&&graph[i][j]!=0){
            p=i;
            q=j;
            min=graph[i][j];
        }
    }
}
    visit[p]=1;
    visit[q]=1;
    int flag1=0;
    int p1,q1,min1,qq=0;
    printf("%d-->%d",arr[p],arr[q]);
    do{
        for(i=0;i<m;i++){
            for(j=0;j<m;j++){
                if(visit[i]==1 && visit[j]==0 && graph[i][j]!=0){
                    if(flag1==0){
                            flag1 = 1;
                            p1=i;
                            q1=j;
                            min1=graph[i][j];
    }else if(flag1==1&&graph[i][j]<min1){
    p1=i;
    q1=j;
    min1=graph[i][j];
}
}
}
}
```

```
    visit[q1]=1;
    flag1=0;
    printf("\n%d-->%d",arr[p1],arr[q1]);
    qq++;
    }while(qq<m-2);
    return 0;
}
```

Результат:



Завдання№6

Розв’язати задачу комівояжера для повного 8-ми вершинного графа методом «іди у найближчий», матриця вагів якого має вигляд:

20)

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

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	**-**	**4**	**6**	**5**	**1**	**2**	**3**	**5**
2	**4**	**-**	**5**	**1**	**5**	**1**	**5**	**1**
3	**6**	**5**	**-**	**5**	**6**	**1**	**5**	**7**
4	**5**	**1**	**5**	**-**	**6**	**4**	**5**	**5**
5	**1**	**5**	**6**	**6**	**-**	**3**	**2**	**2**
6	**2**	**1**	**1**	**4**	**3**	**-**	**2**	**2**
7	**3**	**5**	**5**	**5**	**2**	**2**	**-**	**2**
8	**5**	**1**	**7**	**5**	**2**	**2**	**2**	**-**

	2	**3**	**4**	**15**	**6**	**7**	**8**
2	**-**	**5**	**1**	**5**	**1**	**5**	**1**
3	**5**	**-**	**5**	**6**	**1**	**5**	**7**
4	**1**	**5**	**-**	**6**	**4**	**5**	**5**
15	**5**	**6**	**6**	**-**	**3**	**2**	**2**
6	**1**	**1**	**4**	**3**	**-**	**2**	**2**
7	**5**	**5**	**5**	**2**	**2**	**-**	**2**
8	**1**	**7**	**5**	**2**	**2**	**2**	**-**

	2	3	4	6	157	8
2	-	5	1	1	5	1
3	5	-	5	1	5	7
4	1	5	-	4	5	5
6	1	1	4	-	2	2
157	5	5	2	2	-	2
8	1	7	5	2	2	-

	2	3	1574	6	8
2	-	5	1	1	1
3	5	-	5	1	7
1574	1	5	-	4	5
6	1	1	4	-	2
8	1	7	5	2	-

	15742	3	6	8
15742	-	5	1	1
3	5	-	1	7
6	1	1	-	2
8	1	7	2	-

	3	157426	8
3	-	1	7
154627	1	-	2
8	7	2	-

	1574263	8
1546273	-	7
8	7	-

Шлях:1->5->7->4->2->6->3=1+2+2+2+1+7=15

1 -> 5 -> 7 -> 8 -> 2 -> 4 -> 3 -> 6 (15)
1 -> 6 -> 3 -> 4 -> 2 -> 8 -> 7 -> 5 (15)
2 -> 4 -> 3 -> 6 -> 1 -> 5 -> 7 -> 8 (15)
2 -> 8 -> 7 -> 5 -> 1 -> 6 -> 3 -> 4 (15)
3 -> 4 -> 2 -> 8 -> 7 -> 5 -> 1 -> 6 (15)
3 -> 6 -> 1 -> 5 -> 7 -> 8 -> 2 -> 4 (15)
4 -> 2 -> 8 -> 7 -> 5 -> 1 -> 6 -> 3 (15)
4 -> 3 -> 6 -> 1 -> 5 -> 7 -> 8 -> 2 (15)
5 -> 1 -> 6 -> 3 -> 4 -> 2 -> 8 -> 7 (15)
5 -> 7 -> 8 -> 2 -> 4 -> 3 -> 6 -> 1 (15)
6 -> 1 -> 5 -> 7 -> 8 -> 2 -> 4 -> 3 (15)
6 -> 3 -> 4 -> 2 -> 8 -> 7 -> 5 -> 1 (15)
7 -> 5 -> 1 -> 6 -> 3 -> 4 -> 2 -> 8 (15)
7 -> 8 -> 2 -> 4 -> 3 -> 6 -> 1 -> 5 (15)
8 -> 2 -> 4 -> 3 -> 6 -> 1 -> 5 -> 7 (15)

1-> 2-> 3-> 4-> 5-> 6-> 7-> 8-> 1 (32)
1-> 2-> 3-> 4-> 5-> 6-> 8-> 7-> 1 (30)
1-> 2-> 3-> 4-> 5-> 7-> 6-> 8-> 1 (31)
1-> 2-> 3-> 4-> 5-> 7-> 8-> 6-> 1 (28)
1-> 2-> 3-> 4-> 5-> 8-> 6-> 7-> 1 (29)
1-> 2-> 3-> 4-> 5-> 8-> 7-> 6-> 1 (28)
1-> 2-> 3-> 4-> 6-> 5-> 7-> 8-> 1 (30)
1-> 2-> 3-> 4-> 6-> 5-> 8-> 7-> 1 (28)
1-> 2-> 3-> 4-> 6-> 7-> 5-> 8-> 1 (29)
1-> 2-> 3-> 4-> 6-> 7-> 8-> 5-> 1 (25)
1-> 2-> 3-> 4-> 6-> 8-> 5-> 7-> 1 (27)
1-> 2-> 3-> 4-> 6-> 8-> 7-> 5-> 1 (25)
1-> 2-> 3-> 4-> 7-> 5-> 6-> 8-> 1 (31)
1-> 2-> 3-> 4-> 7-> 5-> 8-> 6-> 1 (27)
1-> 2-> 3-> 4-> 7-> 6-> 5-> 8-> 1 (31)
1-> 2-> 3-> 4-> 7-> 6-> 8-> 5-> 1 (26)
1-> 2-> 3-> 4-> 7-> 8-> 5-> 6-> 1 (28)
1-> 2-> 3-> 4-> 7-> 8-> 6-> 5-> 1 (27)
1-> 2-> 3-> 4-> 8-> 5-> 6-> 7-> 1 (29)
1-> 2-> 3-> 4-> 8-> 5-> 7-> 6-> 1 (27)
1-> 2-> 3-> 4-> 8-> 6-> 5-> 7-> 1 (29)
1-> 2-> 3-> 4-> 8-> 6-> 7-> 5-> 1 (26)
1-> 2-> 3-> 4-> 8-> 7-> 5-> 6-> 1 (28)

1-> 2-> 3-> 4-> 8-> 7-> 6-> 5-> 1 (27)
1-> 2-> 3-> 5-> 4-> 6-> 7-> 8-> 1 (34)
1-> 2-> 3-> 5-> 4-> 6-> 8-> 7-> 1 (32)
1-> 2-> 3-> 5-> 4-> 7-> 6-> 8-> 1 (35)
1-> 2-> 3-> 5-> 4-> 7-> 8-> 6-> 1 (32)
1-> 2-> 3-> 5-> 4-> 8-> 6-> 7-> 1 (33)
1-> 2-> 3-> 5-> 4-> 8-> 7-> 6-> 1 (32)
1-> 2-> 3-> 5-> 6-> 4-> 7-> 8-> 1 (34)
1-> 2-> 3-> 5-> 6-> 4-> 8-> 7-> 1 (32)
1-> 2-> 3-> 5-> 6-> 7-> 4-> 8-> 1 (35)
1-> 2-> 3-> 5-> 6-> 7-> 8-> 4-> 1 (32)
1-> 2-> 3-> 5-> 6-> 8-> 4-> 7-> 1 (33)
1-> 2-> 3-> 5-> 6-> 8-> 7-> 4-> 1 (32)

Код програми:

```
#include <iostream>
#include <stdio.h>
#include <string>
#include <fstream>
using namespace std;
ifstream fin;
string path = "MyFile1";
struct mass
{
 int mas[9];
};
int** input() {
 int count = 8;

 string str;
 str = "";

 fin.open(path);
 int **arr;
 arr = new int*[count];
 for (int i = 0; i < count; i++)
  arr[i] = new int[count];

 for (int i = 0; i < count; i++)
 {
```

```
  for (int j = 0; j < count; j++)
   arr[i][j] = 0;
 }
 for (int i = 0; i < count; i++)
 {
  for (int j = i + 1; j < count; j++)
  {
   getline(fin, str);
   arr[i][j] = atoi(str.c_str());
   arr[j][i] = atoi(str.c_str());
  }
 }
 fin.close();
 return arr;
}
bool comp(int* arr, int count)
{
 int* mas = new int[count];
 for (int i =0; i < count; i++)
 {
  mas[i] = count -i;
 }

 for (int i = 0; i < count; i++)
 {
  if (mas[i] != arr[i])
  {
   return true;
  }
  else
  {
   continue;
  }
 }
 return false;
}
bool povtor(int* mas, int size)
{
 bool k = true;
```

```
 for (int i = 0; i < size; i++)
 {
  for (int j = 0; j < size; j++)
  {
   if (mas[i] == mas[j] && i != j)
   {
    return false;
   }
  }
 }
 return true;
}
int way(int** mat, int* arr)
{
 int count = 0;

 for (int i = 0; i < 7; i++)
 {
  count += mat[arr[i] - 1][arr[i + 1] - 1];
 }
 count += mat[arr[7] - 1][arr[0] - 1];
 return count;
}
int main() {
 int const count = 8;
 int **arr;
 arr = input();
 int var = count - 1;
 bool k = true;
 int *mas = new int[count];

 int* minmas = new int[9];
 int min = 1000;
 int leng = 0;

 int m = 0;
```

```
for (int i = 0; i < count; i++)
{
 mas[i] = 1;
 minmas[i] = 1;
}

while (comp(mas, count))
{
 while (mas[var] != count)
 {
  mas[var]++;

  if (povtor(mas, count))
  {
   leng = way(arr, mas);

   for (int i = 0; i < count; i++)
   {
    cout << mas[i] << "-> ";
   }
   cout << mas[0] << " (" << leng << ") ";
   cout << endl;

   if (leng < min)
   {
    min = leng;
    m = 1;
   }
   if (leng == min)
   {
    m++;
   }
  }
 }

 while (mas[var] == count)
 {
  mas[var] = 1;
  var--;
```

```
 }
 mas[var]++;

 if (povtor(mas, count))
 {
  for (int i = 0; i < count; i++)
  {
   cout << mas[i] << "-> ";
  }
  cout << mas[0] << " (" << leng << ") ";
  cout << endl;

  leng = way(arr, mas);

  if (leng < min)
  {
   min = leng;
   m = 1;
  }
  if (leng == min)
  {
   m++;
  }
 }
 var = count - 1;
}

for (int i = 0; i < count; i++)
{
 mas[i] = 1;
 minmas[i] = 1;
}
mass *rez = new mass[m];
int iter = 0;

while (comp(mas, count))
{
 while (mas[var] != count)
 {
```

```
 mas[var]++;

 if (povtor(mas, count))
 {
  leng = way(arr, mas);

  if (leng == min)
  {
   for (int i = 0; i < count; i++)
   {
    rez[iter].mas[i] = mas[i];
   }
   rez[iter].mas[count] = mas[0];
   iter++;
  }
 }
}
while (mas[var] == count)
{
 mas[var] = 1;
 var--;
}
mas[var]++;

if (povtor(mas, count))
{
 leng = way(arr, mas);
 if (leng == min)
 {
  for (int i = 0; i < count; i++)
  {
   rez[iter].mas[i] = mas[i];
  }
  rez[iter].mas[count] = mas[0];
  iter++;
 }
}
var = count - 1;
}
```

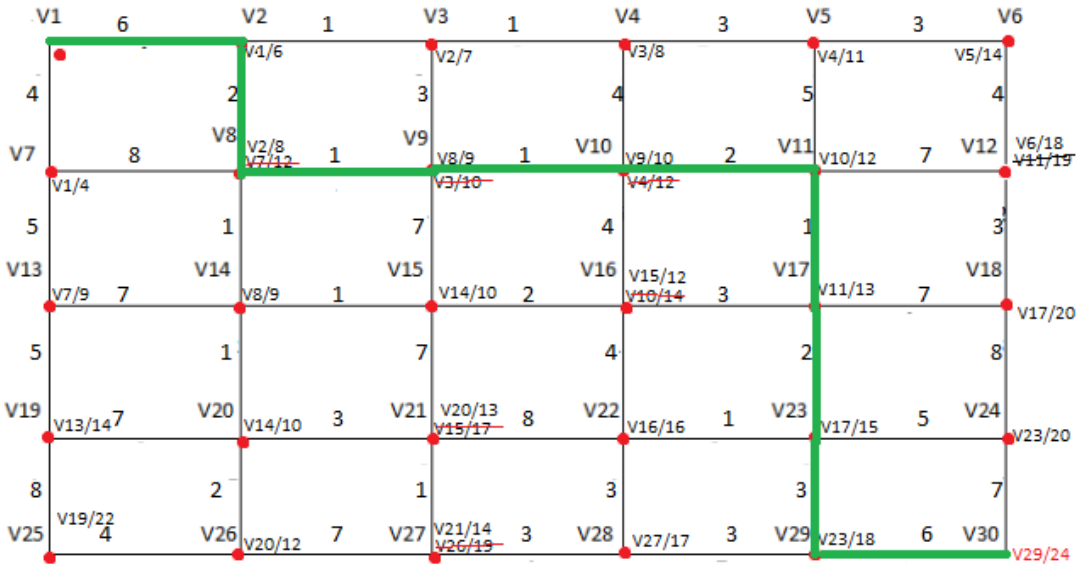
```
 cout << "Ways: " << endl;

 for (int i = 0; i < iter - 1; i++)
 {
  for (int j = 0; j <= count; j++)
  {
   if (j != 0)
   {
    cout << "-> ";
   }
   cout << rez[i].mas[j] << " ";
  }
  cout << endl;
 }
 cout << "Minimal leng = " << min;

 return 0;
}
```

Результати:

```
Консоль отладки Microsoft Visual Studio
8-> 7-> 6-> 5-> 3-> 4-> 1-> 2-> 8 (28)
8-> 7-> 6-> 5-> 3-> 4-> 2-> 1-> 8 (28)
8-> 7-> 6-> 5-> 4-> 1-> 2-> 3-> 8 (34)
8-> 7-> 6-> 5-> 4-> 1-> 3-> 2-> 8 (30)
8-> 7-> 6-> 5-> 4-> 2-> 1-> 3-> 8 (31)
8-> 7-> 6-> 5-> 4-> 2-> 3-> 1-> 8 (31)
8-> 7-> 6-> 5-> 4-> 3-> 1-> 2-> 8 (29)
8-> 7-> 6-> 5-> 4-> 3-> 2-> 1-> 8 (29)
Ways:
1 -> 5 -> 7 -> 8 -> 2 -> 4 -> 3 -> 6 -> 1
1 -> 6 -> 3 -> 4 -> 2 -> 8 -> 7 -> 5 -> 1
2 -> 4 -> 3 -> 6 -> 1 -> 5 -> 7 -> 8 -> 2
2 -> 8 -> 7 -> 5 -> 1 -> 6 -> 3 -> 4 -> 2
3 -> 4 -> 2 -> 8 -> 7 -> 5 -> 1 -> 6 -> 3
3 -> 6 -> 1 -> 5 -> 7 -> 8 -> 2 -> 4 -> 3
4 -> 2 -> 8 -> 7 -> 5 -> 1 -> 6 -> 3 -> 4
4 -> 3 -> 6 -> 1 -> 5 -> 7 -> 8 -> 2 -> 4
5 -> 1 -> 6 -> 3 -> 4 -> 2 -> 8 -> 7 -> 5
5 -> 7 -> 8 -> 2 -> 4 -> 3 -> 6 -> 1 -> 5
6 -> 1 -> 5 -> 7 -> 8 -> 2 -> 4 -> 3 -> 6
6 -> 3 -> 4 -> 2 -> 8 -> 7 -> 5 -> 1 -> 6
7 -> 5 -> 1 -> 6 -> 3 -> 4 -> 2 -> 8 -> 7
7 -> 8 -> 2 -> 4 -> 3 -> 6 -> 1 -> 5 -> 7
8 -> 2 -> 4 -> 3 -> 6 -> 1 -> 5 -> 7 -> 8
Minimal leng = 15
```

Завдання № 7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин *V*0 і *V** .



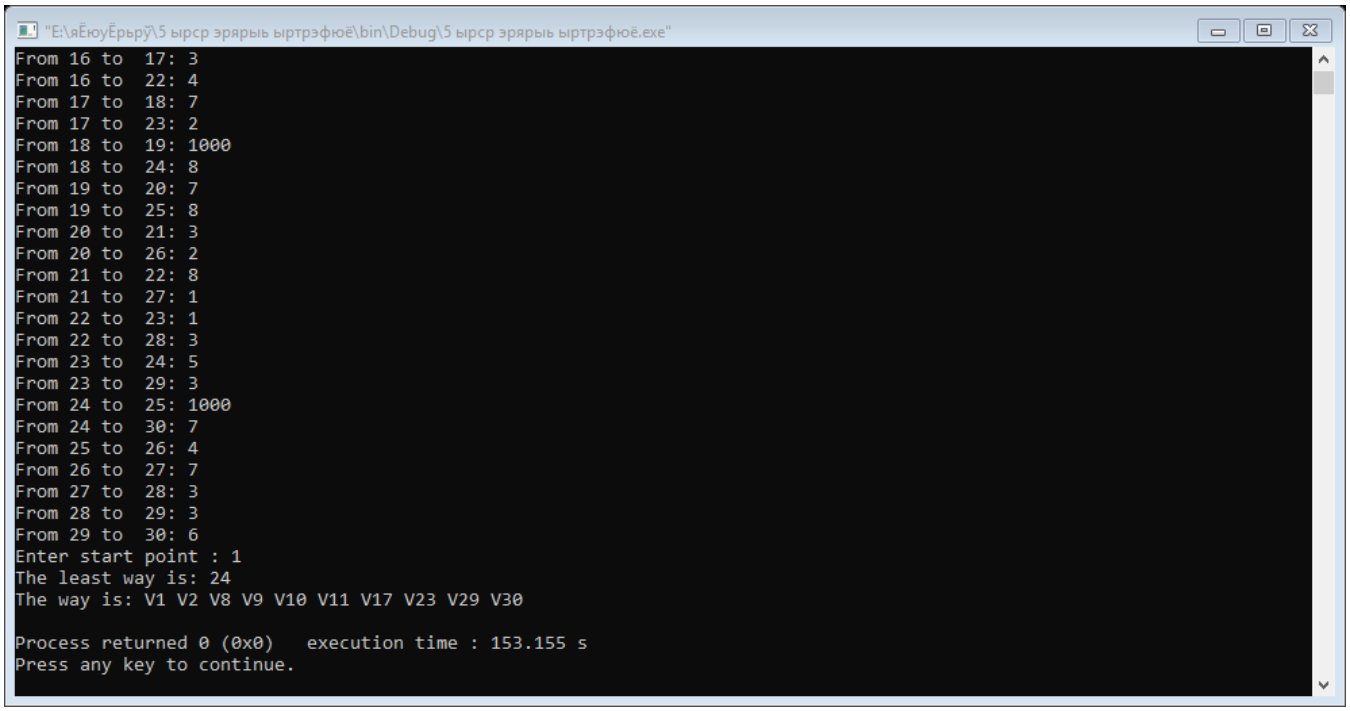
Код програми:

```
#include<iostream>
using namespace std;
int n,i, j, q,dist[40],pred[40],c[40][40];
bool visited[40];
int minDist()
{
   int minimum = 9999, minD;
   for (int v=0; v<n; v++)
      if (!visited[v] && dist[v]<= minimum)
      {
         minimum=dist[v];
         minD=v;
      }
   return minD;
}
void printPath(int j)
{
   if (pred[j]==-1)
      return;
   printPath(pred[j]);
   cout<<"V"<<j+1<< " ";
}
void dijkstra(int c[40][40])
{
   int point;
   cout << "Enter start point : ";
   cin >> point;
   for (int i = 0; i < n; i++)
   {
      pred[0]=-1;
      dist[i]=9999;
      visited[i]=false;
   }
   dist[point-1]=0;
   for (int i=0;i<n - 1;i++)
   {
      int u = minDist();
      visited[u] = true;
```

```
        for (int v=0; v<n; v++)
            if (!visited[v] && c[u][v] &&
                dist[u]+c[u][v] < dist[v])
            {
                pred[v]=u;
                q++;
                dist[v]=dist[u]+c[u][v];
            }
    }
    cout << "The least way is: ";
    cout << dist[29] << endl;
    cout << "The way is: ";
    cout << "V1 ";
    printPath(29);
    cout << endl;
}
int main()
{
    int g1, g2;
    cout << "Enter the number of vertices: ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
        {
            c[i][j] = 0;
        }
    }
    g1=6;
    g2=5;
    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++)
        {
            if (j == i + 1 || j == i + g1) {
                cout << "From " << i+1 << " to  " << j+1 << ": ";
                cin >> c[i][j];
            }
            else {
                c[i][j] = 0;
            }
```

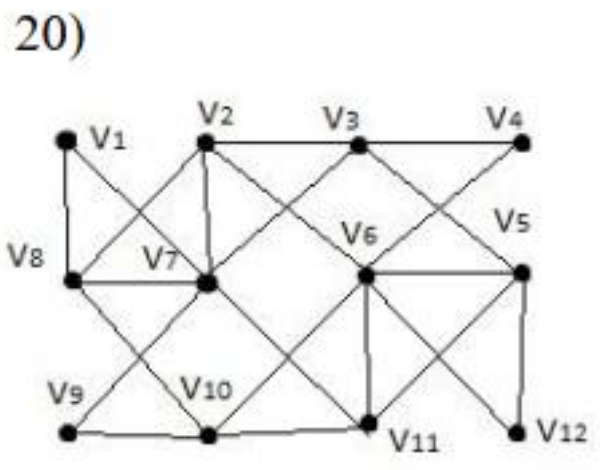
```
        }
    }
    dijkstra(c);
    return 0;
}
```

Результати:



Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів



Флері:V7->V2->V3->V4->V6->V2->V8->V1->V7->V3->V5->V6->V10->V8->V7->V9->V10->V11->V5->V12->V6->V11->V7->V4->V3->V7

1)V7-V2
2)V2-V3
3)V3-V4
4)V4-V6
5)V6-V2
6)V2-V8
8)V8-V1
9)V1-V7
10) V7-V3
11) V3-V5
12) V5-V6
13) V6-V10
14) V10-V8
15) V8-V7
16) V7-V9
17) V9-V10
18)V10-V11
19) V11-V5
20)V5-V12
21)V12-V6
22)V6-V11
23)V11-V7

Код програмної реалізації:

```
#include<iostream>
#include<vector>
using namespace std;
const int n = 12;
int graph[n][n] = {
{0,0,0,0,0,0,1,1,0,0,0,0},
{0, 0,1,0,0,1,1,1,0,0,0,0},
{0,1,0,1,1,0,1,0,0,0,0,0},
{0,0,1,0,0,1,0,0,0,0,0,0},
{0,0,1,0,0,1,0,0,0,0,1,1},
{0,1,0,1,1,0,0,0,0,1,1,1},
{0,1,1,0,0,0,0,1,1,0,1,0},
{1,1,0,0,0,0,1,0,0,1,0,0},
{0,0,0,0,0,0,1,0,0,1,0,0},
{0,0,0,0,0,1,0,1,1,0,1,0},
{0,0,0,0,1,1,1,0,0,0,0,0},
{0,0,0,0,1,1,0,0,0,0,0,0}
};
```

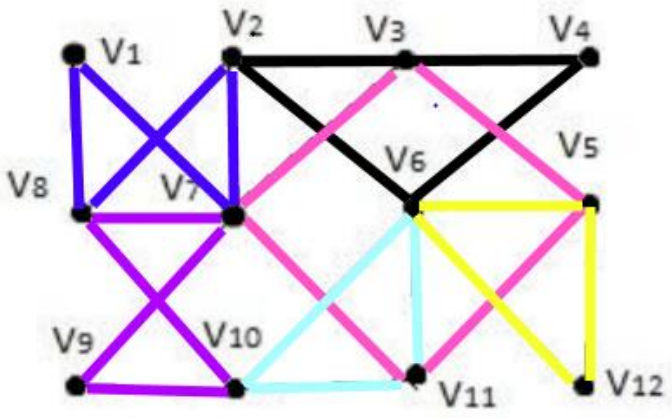
```
int oddGraph[n][n];
int startVertex() {
  for (int i = 0; i < n; i++) {
    int deg = 0;
    for (int j = 0; j < n; j++) {
      if (oddGraph[i][j])
        deg++;    }
    if (deg % 2 != 0)
      return i;
  }
  return 0;}
bool check(int u, int v) {
  int deg = 0;
  for (int i = 0; i < n; i++)
    if (oddGraph[v][i])
      deg++;
  if (deg > 1) {
    return false;
  }
  return true;
}
int edgeCount() {
  int count = 0;
  for (int i = 0; i < n; i++)
    for (int j = i; j < n; j++)
      if (oddGraph[i][j])
        count++;
  return count;
}
void func(int start) {
```

```
  static int edge = edgeCount();
  for (int v = 0; v < n; v++) {
    if (oddGraph[start][v]) {
      if (edge <= 1 || !check(start, v)) {
        cout << start+1 << "-" << v+1 << endl;
        oddGraph[start][v] = oddGraph[v][start] = 0;
        edge--;
        func(v);
      }
    }
  }
}
int main() {
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
      oddGraph[i][j] = graph[i][j];
  cout << "Result: " << endl;
  func(startVertex());
  return 0;
}
```

Результат програми:

```
Result:
7-2
2-3
3-4
4-6
6-2
2-8
8-1
1-7
7-3
3-5
5-6
6-10
10-8
8-7
7-9
9-10
10-11
11-5
5-12
12-6
6-11
11-7

Process returned 0 (0x0)   execution time : 0.437 s
Press any key to continue.
```

Методом елементарних циклів:
Припускаємо, що ейлерів цикл в графі існує (і складається хоча б з однієї вершини). Для пошуку ейлерового циклу скористаємося тим, що ейлерів цикл - це об'єднання всіх простих циклів графа. Отже, завдання - ефективно знайти всі цикли і ефективно об'єднати їх в один.Р озфарбовую всі елементарні в різні кольори та ефективно об’єдную їх в один.



1)V1-V7-V2-V8-V1

2)V8-V7-V9-V10-V8

3)V2-V3-V4-V6-V2

4)V7-V3-V5-V11-V7

5)V10-V6-V11

6)V6-V5-V12

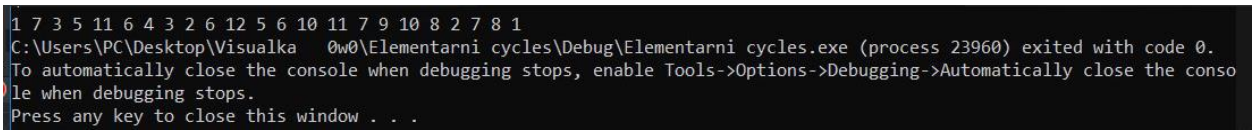
Код програмної реалізації:

```
#include <iostream>
#include <vector>
#include <stack>
#include <algorithm>
#include <list>
using namespace std;
vector < list<int> > graph;
vector <int> deg;
stack<int> head, tail;
int main()
{
  int n, a, x, y;
  cin >> n >> a;
  graph.resize(n + 1);
```

```
  deg.resize(n + 1);
  for (; a--;)
  {
    cin >> x >> y;
    graph[x].push_back(y);
    graph[y].push_back(x);
    ++deg[x];
    ++deg[y];
  }
  if (any_of(deg.begin() + 1, deg.end(), [](int i) {return i & 1;
}))
    cout << "-1";
  else
  {
    head.push(1);
    while (!head.empty())
    {
      while (deg[head.top()])
      {
        int v = graph[head.top()].back();
        graph[head.top()].pop_back();
        graph[v].remove(head.top());
```

```
        --deg[head.top()];
        head.push(v);
        --deg[v];
      }
      while (!head.empty() && !deg[head.top()])
      {
        tail.push(head.top());
        head.pop();
      }
    }
    while (!tail.empty())
    {
      cout << tail.top() << ' ';
      tail.pop();
    }
  }
}
```

Результат програми:



Завдання №9

Спростити формули (привести їх до скороченої ДНФ)

$(x \vee \bar{z})(\bar{y} \vee z)$

$X\bar{Y} \vee X\bar{Z} \vee \overline{YY} \vee \overline{YZ} = X\bar{Y} \vee X\bar{Z} \vee \bar{Y} \vee \overline{YZ}$

2) $X\bar{Y} \vee \bar{Y} = \bar{Y}$(закон поглинання)

$X\bar{Z} \vee \bar{Y} \vee \overline{YZ}$

3) $\bar{Y} \vee \overline{YZ} = \bar{Y}$ (закон поглинання)

$X\bar{Z} \vee \bar{Y}$-сДНФ