МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

Виконав:

студентка групи КН-114

Кміть Христина

Викладач:

Мельникова Н.І.

Львів – 2019р.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

Завдання № 1

Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

2) об’єднання графів,

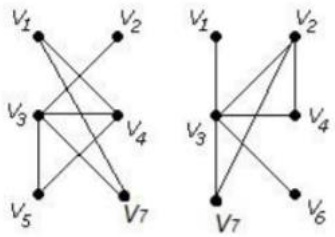
3) кільцеву сумму G1 та G2 (G1+G2),

4) розмножити вершину у другому графі,

5) виділити підграф А - що скадається з 3-х вершин в G1

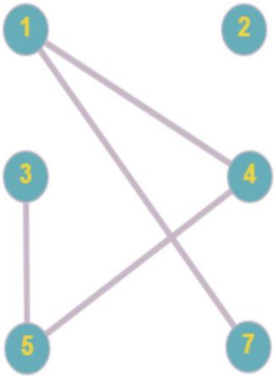
6) добуток графів.

12)

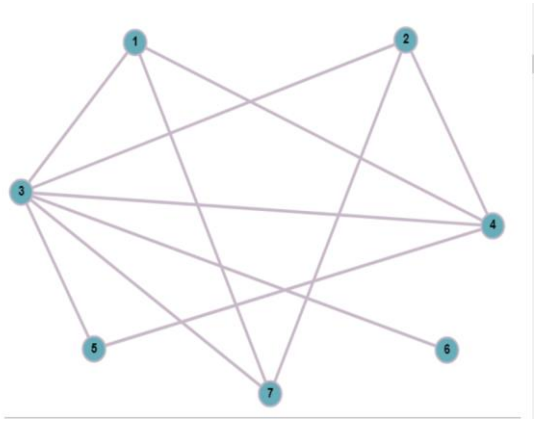


Розв’язок:

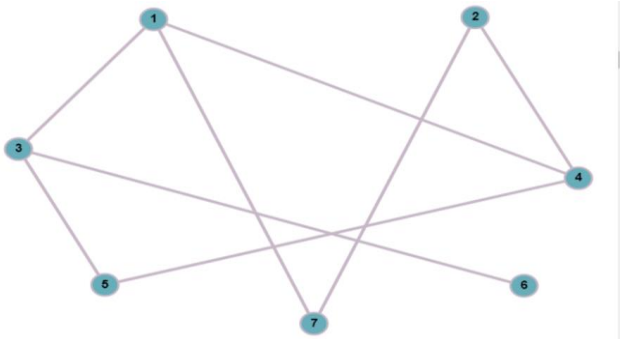
1)



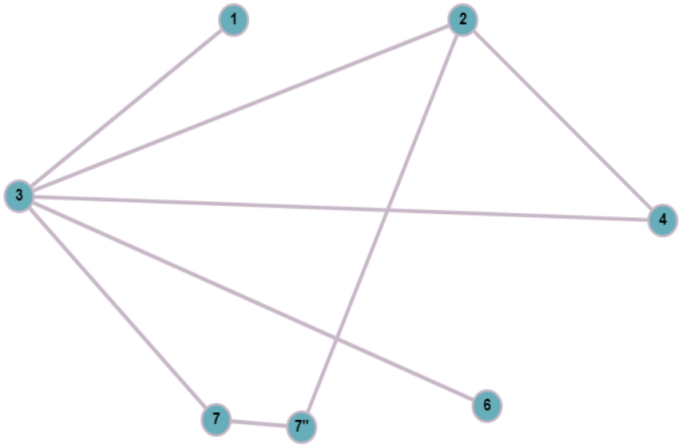
2)



3)

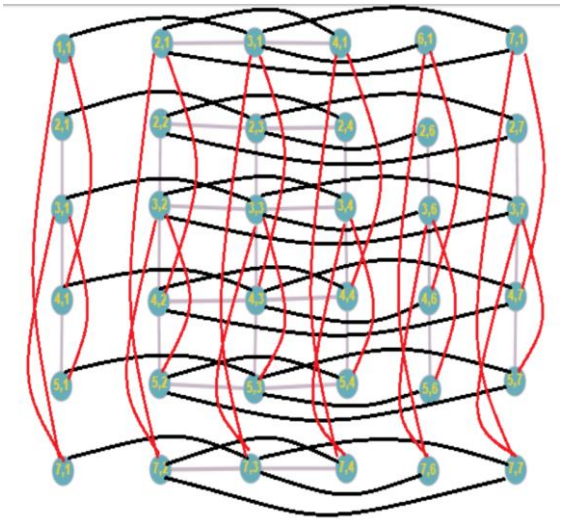


4)



5)

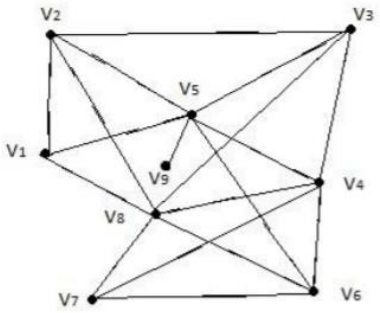
6)



Завдання № 2

Скласти таблицю суміжності для орграфа.

12)



Розв’язок:

x	1	2	3	4	5	6	7	8	9
1	x	1	0	0	1	0	0	1	0
2	1	x	1	0	1	0	0	1	0
3	0	1	x	1	1	0	0	1	0
4	0	0	1	x	1	1	1	1	0
5	1	1	1	1	x	1	0	0	1
6	0	0	0	1	1	x	1	1	0
7	0	0	0	1	0	1	x	1	0
8	1	1	1	1	0	1	1	x	0
9	0	0	0	0	1	0	0	0	x

Завдання № 3

Для графа з другого завдання знайти діаметр.

Діаметром є відстань між вершинами 7 і 9 =4.

Завдання № 4

Для графа з другого завдання виконати обхід дерева вшир:

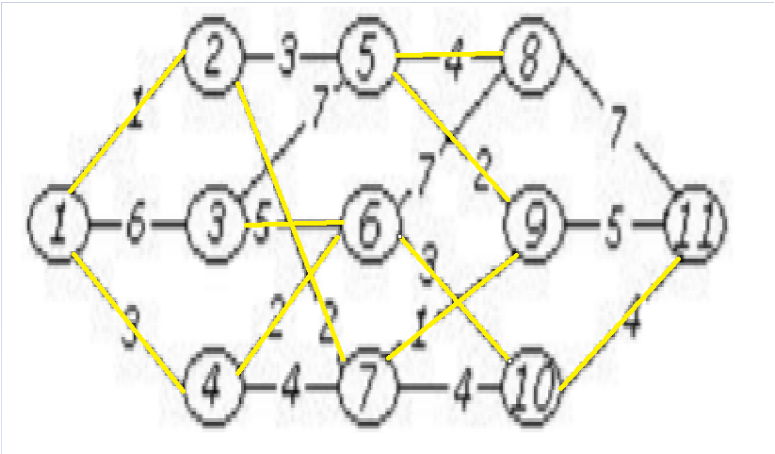
```cpp
#include <iostream>
#include <queue>

using namespace std;

int main()
{
  queue<int> Search;
  int m[9][9] = {
 // 1  2  3  4  5  6  7  8  9
  { 0, 1, 0, 0, 1, 0, 0, 1, 0},
  { 1, 0, 1, 0, 1, 0, 0, 1, 0},
  { 0, 1, 0, 1, 1, 0, 0, 1, 0},
  { 0, 0, 1, 0, 1, 1, 1, 1, 0},
  { 1, 1, 1, 1, 0, 0, 0, 0, 1},
  { 0, 0, 0, 1, 0, 0, 1, 1, 0},
  { 0, 0, 0, 1, 0, 1, 0, 1, 0},
  { 1, 1, 1, 1, 0, 1, 1, 0, 0},
  { 0, 0, 0, 0, 1, 0, 0, 0, 0},

  };
  cout<<"Matrix of your graph:"<<endl;
  for (int i=0; i<9; i++)
  {
    for (int j=0; j<9; j++)
    {
      cout<<m[i][j]<<" ";
    }
    cout<<endl;
  }
  cout<<endl;
  int vertexs[9];
  for (int i = 0; i < 9; i++)
  {
    vertexs[i] = 0;
  }
  Search.push(0);
  while (!Search.empty())
  {
    int v = Search.front();
    Search.pop();
    vertexs[v] = 2; // помічаємо як пройде
    for (int j = 0; j < 11; j++)
    {
      if (m[v][j] == 1 && vertexs[j] == 0)
      {
        Search.push(j);
        vertexs[j] = 1;
      }
    }
    cout << v + 1 <<"  ";
  }
  cout<<endl;
  return 0;
}
```

```
Matrix of your graph:
0 1 0 0 1 0 0 1 0
1 0 1 0 1 0 0 1 0
0 1 0 1 1 0 0 1 0
0 0 1 0 1 1 1 1 0
1 1 1 1 0 0 0 0 1
0 0 0 1 0 0 1 1 0
0 0 0 1 0 1 0 1 0
1 1 1 1 0 1 1 0 0
0 0 0 0 1 0 0 0 0

1  2  5  8  10  3  4  9  6  7
```

Завдання № 5

Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

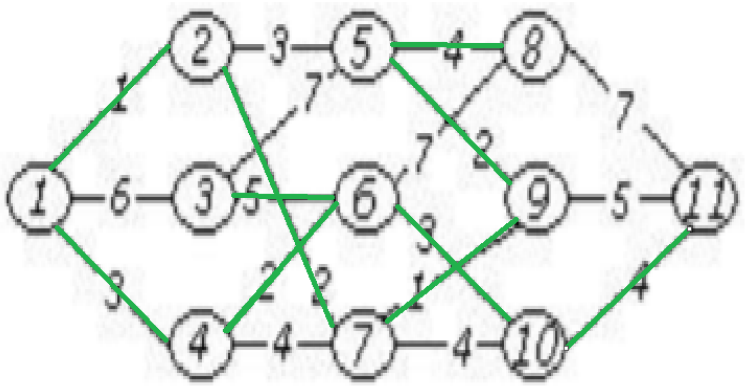
Прима:



```cpp
#include "pch.h"
#include <iostream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "Ukrainian");
    int v, count = 0, min = 0, k, t;
    bool check = false;
    cout << "Кількість вершин графа : ";
    cin >> v;
    int* vershyny = new int[v];
    int** matrix = new int*[v];
    int** rebra = new int*[v - 1];

    for (int i = 0; i < v; i++) {
        matrix[i] = new int[v];
    }

    for (int i = 0; i < v - 1; i++) {
        rebra[i] = new int[2];
    }
    //ввід матриці ваг
    for (int i = 0; i < v; i++) {
        for (int j = 0; j < v; j++) {
            cin >> matrix[i][j];
        }
    }
    //беремо вершину один як початкову
    vershyny[count] = 1;
    count++;

    for (int i = 0; count < v; i++) {
        for (int j = 0; j < count; j++) {
            for (int a = 0; a < v; a++) {
                for (int m = 0; m < count; m++) {
                    //перевірка чи вершина вже була пройде, якщо так, то переходимо до наступної ітерації
                    if (vershyny[m] == a + 1) {
                        check = true;
                    }
                }

                if (check) { check = false; continue; }

                if (min == 0 && matrix[vershyny[j] - 1][a] > 0) {
                    min = matrix[vershyny[j] - 1][a];
                    k = rebra[count - 1][0] = vershyny[j]; t = rebra[count - 1][1] = a + 1;
                    continue;
                }

                if (matrix[vershyny[j] - 1][a] > 0 && matrix[vershyny[j] - 1][a] < min) {
                    min = matrix[vershyny[j] - 1][a];
                    k = rebra[count - 1][0] = vershyny[j]; t = rebra[count - 1][1] = a + 1;
                }
            }
        }
        //Обнуляємо ребро, бо вже не можемо по ньому проходити і маємо опускати в подальшому пошуку
        matrix[k - 1][t - 1] = 0; matrix[t - 1][k - 1] = 0;
        //Додаємо знайдену вершину в масив вершин
        vershyny[count] = t;
        count++;
        min = 0;
    }
    //output
    cout << "V: { ";

    for (int j = 0; j < v; j++) {
        cout << vershyny[j] << ", ";
    }

    cout << "}";
    //і виводимо ребра
    cout << endl << "E:{ ";

    for (int j = 0; j < v - 1; j++) {
        cout << "(" << rebra[j][0] << "," << rebra[j][1] << "),";
    }
    cout << "}";
    return 0;

}
/*
0 1 6 3 0 0 0 0 0 0 0
1 0 0 0 3 0 2 0 0 0 0
6 0 0 0 7 5 0 0 0 0 0
3 0 0 0 0 2 4 0 0 0 0
0 3 7 0 0 0 0 4 2 0 0
0 0 5 2 0 0 0 7 0 3 0
0 2 0 4 0 0 0 0 1 4 0
0 0 0 0 4 7 0 0 0 0 7
0 0 0 0 2 0 1 0 0 0 5
0 0 0 0 0 3 4 0 0 0 4
0 0 0 0 0 0 0 7 5 4 0
*/
```

```
Кількість вершин графа : 11
0 1 6 3 0 0 0 0 0 0 0
1 0 0 0 3 0 2 0 0 0 0
6 0 0 0 7 5 0 0 0 0 0
3 0 0 0 0 2 4 0 0 0 0
0 3 7 0 0 0 0 4 2 0 0
0 0 5 2 0 0 0 7 0 3 0
0 2 0 4 0 0 0 0 1 4 0
0 0 0 0 4 7 0 0 0 0 7
0 0 0 0 2 0 1 0 0 0 5
0 0 0 0 0 3 4 0 0 0 4
0 0 0 0 0 0 0 7 5 4 0
V: { 1, 2, 7, 9, 5, 4, 6, 10, 8, 11, 3, }
E:{ (1,2),(2,7),(7,9),(9,5),(1,4),(4,6),(6,10),(5,8),(10,11),(6,3),}
```

Краскала:




```cpp
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
int main() {
int m=18,n=11;
vector < pair < int, pair<int,int> > > g (m); // вага - вершина 1
cout<<"Input your graph(1 is weight)(2 & 3 are vertexes:)\n";
for(int i=0;i<m;i++)
cin>>g[i].first>> g[i].second.first>>g[i].second.second;

int cost = 0,l=0;
vector < pair<int,int> > res;
sort (g.begin(), g.end());
vector<int> tree_id (n);
for (int i=0; i<n-1; ++i)
tree_id[i] = i;
for (int i=0; i<m; ++i)
{
int a = g[i].second.first, b = g[i].second.second; l = g[i].first;
if (tree_id[a] != tree_id[b])
{
cost += l;
res.push_back (make_pair (a, b));
int old_id = tree_id[a], new_id = tree_id[b];
for (int j=0; j<n; ++j)
if (tree_id[j] == old_id)
tree_id[j] = new_id;
}
}
cout<<"Vertexes of graph are\n";
for (int i = 0; i < n -1 ; i++)
cout << res[i].first << " " << res[i].second<<endl;
cout<<"Sum of graph is="<<cost;
}
```

```
Input your graph(1 is weight)(2 & 3 are vertexes:)
1 1 2
3 1 4
6 1 3
3 2 5
2 2 7
7 3 5
5 3 6
2 4 6
4 4 7
7 5 8
1 5 9
4 6 8
3 6 10
2 7 9
4 7 10
7 8 11
4 9 11
5 10 11
Vertexes of graph are
1 2
5 9
2 7
4 6
7 9
1 4
6 10
6 8
9 11
3 6
Sum of graph is=27
```

Завдання № 6

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

12)

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

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

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

	215	3	4	6	7	8
215	-	1	5	1	1	1
3	1	-	1	3	2	1
4	5	1	-	5	7	5
6	1	3	5	-	5	6
7	1	2	7	5	-	1
8	1	1	5	6	1	-

	3215	4	6	7	8
3215	-	1	3	2	1
4	1	-	5	7	5
6	3	5	-	5	6
7	2	7	5	-	1
8	1	5	6	1	-

	43215	6	7	8
43215	-	5	7	5
6	5	-	5	6
7	7	5	-	1
8	5	6	1	-

	643215	7	8
643215	-	5	6
7	5	-	1
8	6	1	-

	7643215	8
7643215	-	1
8	1	-

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

7643125		8
7643215	-	1
8	1	-

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

	1724635	8
1724635	-	1
8	1	-

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

	3172645	8
3172645	-	1
8	1	-

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

	5612475	8
5612475	-	1
8	1	-

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

	6512473	8
6512473	-	1
8	1	-

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

	7516243	8
7516243	-	1
8	1	-

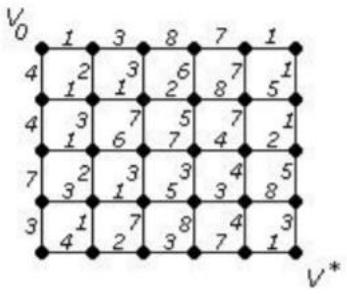
	1	2	3	4	5	6	7	8
1	-	5	6	5	4	4	5	5
2	5	-	1	5	1	1	1	1
3	6	1	-	1	1	3	2	1
4	5	5	1	-	5	5	7	5
5	4	1	1	5	-	3	2	5
6	4	1	3	5	3	-	5	6
7	5	1	2	7	2	5	-	1
8	5	1	1	5	5	6	1	-

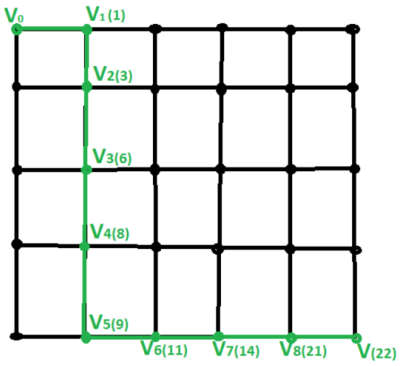
	2178534	6
218534	-	5
6	5	-

Завдання № 7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V * .

12)





Розв’язок:

```
ConsoleApplication5                                   (Global Scope)
#include "pch.h"
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#define SIZE 30
int main()
{
    int a[SIZE][SIZE]; // матриця зв'язків
    int d[SIZE]; // мінімальна відстань
    int v[SIZE]; //відвідані вершини
    int temp, minindex, min;
    int begin_index = 0;
    system("chcp 1251");
    system("cls");
    // Ініціалізація матриці зв'язків
    for (int i = 0; i < SIZE; i++)
    {
        a[i][i] = 0;
        for (int j = i + 1; j < SIZE; j++) {
            printf("Введіть відстань [%d-%d] : ", i + 1, j + 1);
            scanf("%d", &temp);
            a[i][j] = temp;
            a[j][i] = temp;
        }
    }
    //Вивід матриці зв'язків
    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < SIZE; j++)
            printf("%5d ", a[i][j]);
        printf("\n");
    }
    //Ініціалізація вершин і відстаней
    for (int i = 0; i < SIZE; i++)
    {
        d[i] = 10000;
        v[i] = 1;
    }
    d[begin_index] = 0;
    do {
        minindex = 10000;
        min = 10000;
        for (int i = 0; i < SIZE; i++)
        { // Якщо вершину ще не оминули й вага менше min
            if ((v[i] == 1) && (d[i] < min))
            {
                min = d[i];
                minindex = i;
            }
        }
        // Додаємо знайдений мінімальна вага до поточного в
        if (minindex != 10000)
        {
            for (int i = 0; i < SIZE; i++)
            {
                if (a[minindex][i] > 0)
                {
                    temp = min + a[minindex][i];
                    if (temp < d[i])
```

ConsoleApplication5.cpp

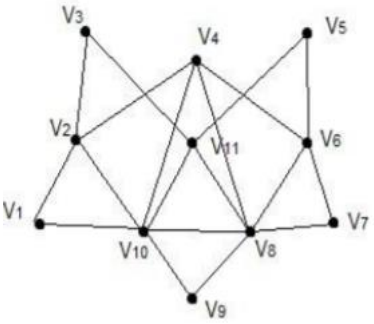
```
ConsoleApplication5                                   (Global Scope)
                    if (temp < d[i])
                    {
                        d[i] = temp;
                    }
                }
            }
            v[minindex] = 0;
        }
    } while (minindex < 10000);
    // Висновок найкоротших відстаней до вершин
    printf("\nНайкоротші відстані до вершин: \n");
    for (int i = 0; i < SIZE; i++)
        printf("%5d ", d[i]);

    // Відновлення шляху
    int ver[SIZE]; // масив відвіданих вершин
    int end = 28; // індекс кінцевої вершини = 5 - 1
    ver[0] = end + 1; // початковий елемент - кінцева вершина
    int k = 1; // індекс попередньої вершини
    int weight = d[end]; // вага кінцевої вершини

    while (end != begin_index) // поки не дійшли до початкової вершини
    {
        for (int i = 0; i < SIZE; i++) // переглядаємо всі вершини
            if (a[end][i] != 0)   // якщо зв'язок є
            {
                int temp = weight - a[end][i]; // визначаємо вагу шляху з попередньої вершини
                if (temp == d[i]) // якщо вага припала на розрахованим
                {                 // значить з цієї вершини і був перехід
                    weight = temp; // зберігаємо новий вага
```

```
    int k = 1; // індекс попередньої вершини
    int weight = d[end]; // вага кінцевої вершини

    while (end != begin_index) // поки не дійшли
    {
        for (int i = 0; i < SIZE; i++) // перегля
            if (a[end][i] != 0)   // якщо зв'язок
            {
                int temp = weight - a[end][i]; //
                if (temp == d[i]) // якщо вага пр
                {                 // значить з ці
                    weight = temp; // зберігаємо
                    end = i;       // зберігаємо
                    ver[k] = i + 1; // і записуєм
                    k++;
                }
            }
    }
    // Вивід шляху (початкова вершина виявилася в
    printf("\nВивід найкоротшого шляху\n");
    for (int i = k - 1; i >= 0; i--)
        printf("%3d ", ver[i]);
    getchar(); getchar();
    return 0;
}
```

Завдання № 8

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

12)

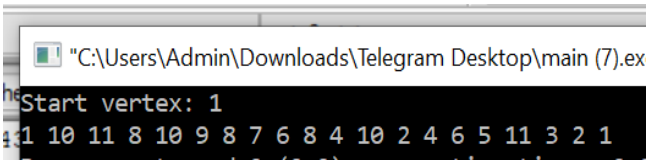


а)

```
#include <iostream>
#include <cstdio>

#define N 12
#define STACK_SIZE 100
using namespace std;

int G[N][N] {
{0,1,0,0,0,0,0,0,0,1,0},
{1,0,1,1,0,0,0,0,0,1,0},
{0,1,0,0,0,0,0,0,0,0,1},
{0,1,0,0,0,1,0,1,0,1,0},
{0,0,0,0,0,1,0,0,0,0,1},
{0,0,0,0,1,1,0,1,1,0,0},
{0,0,0,0,0,0,1,0,1,0,0},
{0,0,0,1,0,1,1,0,1,1,1},
{0,0,0,0,0,0,0,1,0,1,0},
{1,1,0,1,0,0,0,1,1,0,1},
{0,0,1,0,1,0,0,1,0,1,0},
};

int k;
int Stack[STACK_SIZE];
```

```
void Search(int v)
{
    int i;
    for(i = 0; i < N; i++)
        if(G[v][i])
        {
            G[v][i] = G[i][v] = 0;
            Search(i);
        }
    Stack[++k] = v+1;
}
int main()
{
    int T, p, q, s;
    int j, vv;

    T = 1;
    for(p = 0; p < N; p++)
    {
        s = 0;
        for(q = 0; q < N; q++)
        {
            s += G[p][q];
        }
```

```
43      {
44          s = 0;
45          for(q = 0; q < N; q++)
46          {
47              s += G[p][q];
48          }
49          if(s%2) T = 0;
50      }
51      k = -1;
52      cout << "Start vertex: ";
53      cin >> vv;
54      vv-=1;
55      if(T)
56      {
57          Search (vv);
58          for(j = 0; j <= k; j++)
59              cout << Stack[j] << " ";
60      }
61      else
62          cout << "it is not Eulerian graph\n";
63      return 0;
64  }
```



б)

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

12. $\bar{x}y \vee x\bar{y}\bar{z}$

СДНФ:

⌐xy⌐z∪⌐xyz∪x⌐y⌐z