Assignment# 1

Implementation of Floyd-Warshall Algorithm by C#

```
using System;

namespace WarshallAlgo {
   class Program {
      static void Algo(int[,] w, int num) {
         double[,] dis = new double[num, num];
         for (int i = 0; i < num; i++) {
            for (int j = 0; j < num; j++) {
               dis[i, j] = double.PositiveInfinity;
            }
         }

         for (int i = 0; i < w.GetLength(0); i++) {
            dis[w[i, 0] - 1, w[i, 1] - 1] = w[i, 2];
         }

         int[,] next = new int[num, num];
         for (int i = 0; i < num; i++) {
            for (int j = 0; j < num; j++) {
               if (i != j) {
                  next[i, j] = j + 1;
               }
            }
         }

         for (int k = 0; k < num; k++) {
            for (int i = 0; i < num; i++) {
               for (int j = 0; j < num; j++) {
                  if (dis[i, k] + dis[k, j] < dis[i, j]) {
                     dis[i, j] = dis[i, k] + dis[k, j];
                     next[i, j] = next[i, k];
                  }
               }
            }
         }

         Output(dis, next);
      }
```

```
    static void Output(double[,] dis, int[,] next) {
        Console.WriteLine("pair     dis    result");
        for (int i = 0; i < next.GetLength(0); i++) {
            for (int j = 0; j < next.GetLength(1); j++) {
                if (i != j) {

                    int u = i + 1;
                    int v = j + 1;
                    string result = string.Format("{0} goes to {1}    {2,2:G}     {3}", u, v, dis[i, j], u);
                    do {
                        u = next[u - 1, v - 1];
                        result += " -> " + u;
                    } while (u != v);
                    Console.WriteLine(result);
                }
            }
        }
    }

    static void Main(string[] args) {
        int[,] w = { { 2, 1, -3 }, { 4, 2, 3 }, { 1, 2, 3 }, { 3, 4, 2 }, { 5, 2, -2 } };
        int num = 7;

        Algo(w, num);
    }
  }
}
```

Output

```
pair     dis    result
1 goes to 2     3     1 -> 2
1 goes to 3    Infinity     1 -> 3
1 goes to 4    Infinity     1 -> 4
1 goes to 5    Infinity     1 -> 5
1 goes to 6    Infinity     1 -> 6
1 goes to 7    Infinity     1 -> 7
2 goes to 1    -3     2 -> 1
2 goes to 3    Infinity     2 -> 3
2 goes to 4    Infinity     2 -> 4
2 goes to 5    Infinity     2 -> 5
2 goes to 6    Infinity     2 -> 6
2 goes to 7    Infinity     2 -> 7
3 goes to 1     2     3 -> 4 -> 2 -> 1
3 goes to 2     5     3 -> 4 -> 2
3 goes to 4     2     3 -> 4
3 goes to 5    Infinity     3 -> 5
3 goes to 6    Infinity     3 -> 6
3 goes to 7    Infinity     3 -> 7
4 goes to 1     0     4 -> 2 -> 1
4 goes to 2     3     4 -> 2
4 goes to 3    Infinity     4 -> 3
4 goes to 5    Infinity     4 -> 5
4 goes to 6    Infinity     4 -> 6
4 goes to 7    Infinity     4 -> 7
5 goes to 1    -5     5 -> 2 -> 1
5 goes to 2    -2     5 -> 2
5 goes to 3    Infinity     5 -> 3
5 goes to 4    Infinity     5 -> 4
5 goes to 6    Infinity     5 -> 6
5 goes to 7    Infinity     5 -> 7
6 goes to 1    Infinity     6 -> 1
6 goes to 2    Infinity     6 -> 2
6 goes to 3    Infinity     6 -> 3
6 goes to 4    Infinity     6 -> 4
6 goes to 5    Infinity     6 -> 5
6 goes to 7    Infinity     6 -> 7
7 goes to 1    Infinity     7 -> 1
7 goes to 2    Infinity     7 -> 2
7 goes to 3    Infinity     7 -> 3
7 goes to 4    Infinity     7 -> 4
7 goes to 5    Infinity     7 -> 5
7 goes to 6    Infinity     7 -> 6
>
```