S.No: 10	Exp. Name: ***Floyd - Warshall's Algorithm***	Date:2025-09-18

Aim:

Implement the Floyd-Warshall algorithm in C for finding the shortest distances between all pairs of vertices in a weighted directed graph. Prompt the user to input the number of vertices (N) and edges (E), and then accept edge information (source, destination, and weight) to build the adjacency matrix.

Source Code:

Warshall.c

```
#include <stdio.h>
#define INF 99999
#define MAX_N 20 // Maximum value for N
#include <limits.h>

void floydWarshall(int dist[][10],int N)
{
   int i, j, k;

   for(k =0; k< N; k++)
      {
         for(i = 0; i<N; i++)
            {
               for(j =0 ; j< N;j++)
                  {
                     if(dist[i][k]!= INF && dist[k][j] != INF && dist[i][k] + dist[k]
[j] < dist[i][j]){
                        dist[i][j]= dist[i][k]+dist[k][j];
                     }
                  }
            }
      }
}

void printSolution(int dist[][10], int N)
{
   printf("The following matrix shows the shortest distances between all pairs of the
vertices.\n");
   for(int i = 0; i<N;i++)
      {
         for(int j = 0 ; j< N; j++)
            {
               if(dist[i][j] == INF){
                  printf("%5s","INF");
               }else{
                  printf("%5d",dist[i][j]);
               }
            }
         printf("\n");
      }
}

int main(){
   int N,E;
```

```
    printf("Enter the number of vertices : ");
    scanf("%d",&N);

    int dist[10][10];

    for(int i = 0; i< N; i++)
       {
          for(int j =  0; j< N;j++)
             {
                if(i == j){
                   dist[i][j] = 0;
                }else{
                   dist[i][j] = INF;
                }
             }
       }
    printf("Enter the number of edges : ");
    scanf("%d",&E);

    for(int i = 0 ;i<E;i++)
       {
          int src, dest, weight;
          printf("Enter source : ");
          scanf("%d",&src);
          printf("Enter destination : ");
          scanf("%d",&dest);
          printf("Enter weight : ");
          scanf("%d",&weight);
          src -= 1;
          dest -= 1;

          dist[src][dest] = weight;
       }

    floydWarshall(dist,N);

    printSolution(dist,N);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of vertices : 4
Enter the number of edges : 5
Enter source : 1
Enter destination : 2
Enter weight : 4
Enter source : 1
Enter destination : 4
Enter weight : 10

```
Enter source : 1
Enter destination : 3
Enter weight : 6
Enter source : 2
Enter destination : 4
Enter weight : 5
Enter source : 3
Enter destination : 4
Enter weight : 2
The following matrix shows the shortest distances between all pairs of the vertices.
   0    4    6    8
 INF    0  INF    5
 INF  INF    0    2
 INF  INF  INF    0
```

Test Case - 2

User Output

```
Enter the number of vertices : 5
Enter the number of edges : 6
Enter source : 1
Enter destination : 2
Enter weight : 2
Enter source : 1
Enter destination : 5
Enter weight : 3
Enter source : 2
Enter destination : 4
Enter weight : 4
Enter source : 2
Enter destination : 3
Enter weight : 7
Enter source : 4
Enter destination : 3
Enter weight : 2
Enter source : 5
Enter destination : 4
Enter weight : 1
The following matrix shows the shortest distances between all pairs of the vertices.
   0    2    6    4    3
 INF    0    6    4  INF
 INF  INF    0  INF  INF
 INF  INF    2    0  INF
 INF  INF    3    1    0
```

Test Case - 3

User Output

```
Enter the number of vertices : 4
Enter the number of edges : 5
Enter source : 1
Enter destination : 2
```

```
Enter weight : 4
Enter source : 3
Enter destination : 2
Enter weight : 5
Enter source : 4
Enter destination : 1
Enter weight : 1
Enter source : 4
Enter destination : 2
Enter weight : 3
Enter source : 4
Enter destination : 3
Enter weight : 8
The following matrix shows the shortest distances between all pairs of the vertices.
    0    4  INF  INF
  INF    0  INF  INF
  INF    5    0  INF
    1    3    8    0
```

Test Case - 4

User Output

```
Enter the number of vertices : 4
Enter the number of edges : 6
Enter source : 1
Enter destination : 2
Enter weight : 1
Enter source : 1
Enter destination : 4
Enter weight : 3
Enter source : 2
Enter destination : 3
Enter weight : 6
Enter source : 3
Enter destination : 1
Enter weight : -2
Enter source : 4
Enter destination : 2
Enter weight : 5
Enter source : 4
Enter destination : 3
Enter weight : 10
The following matrix shows the shortest distances between all pairs of the vertices.
    0    1    7    3
    4    0    6    7
   -2   -1    0    1
    8    5   10    0
```