ASSIGNMENT NO:8 **DATE:05/04/2016**

PROGRAM TITLE:Find the Minimum Path of all pair of vertices in a Simple Graph using Floyd-Warshall's Algorithm.

PROGRAM ALGORITHM:

```
Floyd_Warshall(int n,int w[1...n][1...n])
{
      array d[1...n][1...n]
      for i=1 to n do
            for j=1 to n do
                  d[i][j]=w[i][j]
                  pred[i][j]=NULL
      for k=1 to n do
            for i=1 to n do
                  for j=1 to n do
                        if(d[i][k]+d[k][j]<d[i][j])
                              d[i][j]=d[i][k]+d[k][j]
                              pred[i][j]=k
}
```

PROGRAM CODE:

```
//C Program to find smallest path between all pair of vertices using Floyd
Warshall
#include <stdio.h>
#include <stdlib.h>
#define datatype int
void create_graph(datatype **graph,int n)
{
      int i,j,x;
      datatype w;
      for(i=0;i<n;i++)
      {
            for(j=0;j<n;j++)
            {
                  graph[i][j]=9999;
            }
      }
      printf("\tIdentify the adjoining vertices:");
      for(i=0;i<n;i++)
      {
            printf("\n\tEnter the adjoining vertices of %d.Enter -99 to
stop::",i);
            for(j=0;j<n;j++)
            {
                  scanf("%d",&x);
                  if(x==-99)
                        break;
                  else
                  {
                        if(graph[x][i]==9999)
                        {
                              printf("\tEnter corresponding weight::");
                              scanf("%d",&w);
                              graph[i][x]=w;
```

```
                        }
                        else
                              graph[i][x]=graph[x][i];
                        printf("\tEnter next adjoining vertex::");
                  }
            }
      }
}
void print_graph(datatype **graph,int n)
{
      int i,j;
      printf("\n\tThe Adjacency Matrix of the Graph is::");
      for(i=-1;i<n;i++)
      {
            printf("\n");
            for(j=-1;j<n;j++)
            {
                  if(i==-1)
                        printf("%d",j);
                  else if(j==-1)
                        printf("%d",i);
                  else
                        printf("%d",graph[i][j]);
                  printf("\t");
            }
      }
      printf("\n");
}
void print_path(int **prev,int i,int j)
{
      if(prev[i][j]==-1)
            printf("-%d-%d",i,j);
      else
      {
            print_path(prev,i,prev[i][j]);
            print_path(prev,prev[i][j],j);
      }
}
void floyd_warshall(datatype **graph,int n)
{
      int i,j,k,s,dest;
      char ch='y';
      datatype **d= (datatype**)malloc(n*sizeof(datatype*));
      datatype **prev= (int**)malloc(n*sizeof(int*));
      for(i=0;i<n;i++)
      {
            d[i]=(datatype*)malloc(n*sizeof(datatype));
            prev[i]=(int*)malloc(n*sizeof(int));
      }
      for(i=0;i<n;i++)
      {
            for(j=0;j<n;j++)
            {
                  d[i][j]=graph[i][j];
                  prev[i][j]=-1;
            }
      }
```

```
	for(k=1;k<n;k++)
	{
		for(i=0;i<n;i++)
		{
			for(j=0;j<n;j++)
			{
				if((d[i][k]+d[k][j])<d[i][j])
				{
					d[i][j]=d[i][k]+d[k][j];
					prev[i][j]=k;
				}
			}
		}
	}
	printf("\n\tThe distances are::");
	for(i=-1;i<n;i++)
	{
		printf("\n");
		for(j=-1;j<n;j++)
		{
			if((i==-1)&&(j==-1))
				printf("From|To");
			else if(i==-1)
				printf("%d",j);
			else if(j==-1)
				printf("%d",i);
			else
				printf("%d",d[i][j]);
			printf("\t");
		}
	}
	printf("\n");
	do
	{
		printf("\n\tDo you want to display a path(y/n)::");
		getchar();
		scanf("%c",&ch);
		if(ch=='n'||ch=='N')
			break;
		printf("\tEnter source::");
		scanf("%d",&s);
		printf("\tEnter destination:");
		scanf("%d",&dest);
		if(d[s][dest]!=9999)
		{
			printf("\tThe path from %d to %d is::",s,dest);
			print_path(prev,s,dest);
		}
		else
			printf("\n\tNo such path exists.");
	}
	while(ch=='y'||ch=='Y');
}
int main()
{
	int n,i;
	printf("\n\tEnter the number of vertices::");
```

```
	scanf("%d",&n);
	datatype **graph= (datatype**)malloc(n*sizeof(datatype*));
	for(i=0;i<n;i++)
	{
		graph[i]=(datatype*)malloc(n*sizeof(datatype));
	}
	create_graph(graph,n);
	print_graph(graph,n);
	floyd_warshall(graph,n);
	return 0;
}
```

OUTPUT:

```
	Enter the number of vertices::6
	Identify the adjoining vertices:
	Enter the adjoining vertices of 0.Enter -99 to stop::1
	Enter corresponding weight::7
	Enter next adjoining vertex::2
	Enter corresponding weight::9
	Enter next adjoining vertex::5
	Enter corresponding weight::14
	Enter next adjoining vertex::-99

	Enter the adjoining vertices of 1.Enter -99 to stop::0
	Enter next adjoining vertex::2
	Enter corresponding weight::10
	Enter next adjoining vertex::3
	Enter corresponding weight::15
	Enter next adjoining vertex::-99

	Enter the adjoining vertices of 2.Enter -99 to stop::0
	Enter next adjoining vertex::1
	Enter next adjoining vertex::5
	Enter corresponding weight::10
	Enter next adjoining vertex::3
	Enter corresponding weight::11
	Enter next adjoining vertex::-99

	Enter the adjoining vertices of 3.Enter -99 to stop::2
	Enter next adjoining vertex::1
	Enter next adjoining vertex::4
	Enter corresponding weight::6
	Enter next adjoining vertex::-99

	Enter the adjoining vertices of 4.Enter -99 to stop::3
	Enter next adjoining vertex::5
	Enter corresponding weight::9
	Enter next adjoining vertex::-99

	Enter the adjoining vertices of 5.Enter -99 to stop::0
	Enter next adjoining vertex::2
	Enter next adjoining vertex::4
	Enter next adjoining vertex::-99

	The Adjacency Matrix of the Graph is::
-1	0	1	2	3	4	5
```

```
0     9999  7     9     9999  9999  14
1     7     9999  10    15    9999  9999
2     9     10    9999  11    9999  10
3     9999  15    11    9999  6     9999
4     9999  9999  9999  6     9999  9
5     14    9999  10    9999  9     9999

      The distances are::
From|To    0     1     2     3     4     5
0     14    7     9     20    23    14
1     7     20    10    15    21    20
2     9     10    20    11    17    10
3     20    15    11    12    6     15
4     23    21    17    6     12    9
5     14    20    10    15    9     18

      Do you want to display a path(y/n)::y
      Enter source::1
      Enter destination:5
      The path from 1 to 5 is::-1-2-2-5
      Do you want to display a path(y/n)::y
      Enter source::0
      Enter destination:4
      The path from 0 to 4 is::-0-5-5-4
      Do you want to display a path(y/n)::n
```

DISCUSSION:

1. We use Adjacency Matrix for storing the graph.
2. Complexity is $\Theta(|V|^3)$
3. The default weight is set arbitrarily high so that it is not considered in the computations.
4. The program works for both directed and undirected graph.