VAKAMALLA KEERTHI PRIYA
(1BM19CS176)

2. Implement Warshall's algorithm using dynamic programming.

Modification: By using path matrix obtained detect the cycle in the graph.

```
#include<stdio.h>
#define V 4
void printSolution(int dist[][V]);
int min(int i, int j)
{
  if(i<j)
  return i;
  return j;
}
void floyd(int A[][4])
{
  int i, j, k, P[4][4];
  for(i=0; i<4; i++)
  for(j=0; j<4; j++)
  P[i][j]=A[i][j];
for(k=0; k<4; k++)
 for(i=0; i<4; i++)
  for(j=0; j<4; j++)
    if(P[i][k]==1 && P[k][j]==1)
      P[i][j]=1;
```

```
    PrintSolution(P);
}
void printSolution(int dist[][V])
{
    printf("The following matrix shows the shortest
            distances between every pair of vertices\n");
for(int i=0; i<V; i++)
    {
        for(int j=0; j<V; j++)
        {
            printf("%.1d", dist[i][j]);
        }
        printf
    }
}
int main()
{
    int graph[V][V] = {
                        {0,1,0,0},
                        {0,0,0,1},
                        {0,0,0,0},
                        {1,0,1,0}
                        };
    printf("\n");
    floyd(graph);
}
```

Modification: By using path matrix obtained detect the cycle in the graph.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
int i, j;
Printf("Enter the Adjacency matrix (1/0) \n");
    for(i=1; i<=n; i++)
    for(j=1; j<=n; j++)
    scanf("%d", &A[i][j]);

Printf("The Depth first search Traversal : \n");
  DFS();

  for(i=1; i<=n; i++)
  Printf("%c, %d \t", 'a'+ seq[i]-1, i);

 if (connected && acyclic)
Printf("\n it is a connected, Acyclic graph");
 if (!connected && acyclic)
Printf("\n it is Not-connected, Acyclic graph");
 if (connected && !acyclic)
Printf("\n Graph is connected, cyclic graph");
if (!connected && !acyclic)
Printf("\n it is not-connected, cyclic graph");
```

```
}
void DFS()
{int i;
 for(i=1; i<=n; i++)
  if(!visited[i])
   {
   if (i>1) connected=0;
    DFSearch(i);
   }
}
void DFSearch(int cur)
   {
   int i, j;
  visited[cur]=++count;
   seq[count]= cur;
   for(i=1; i<count-1; i++)
     if (A[cur][seq[i]])
     acyclic=0;
  for(i=1; i<=n; i++)
    if(A[cur][i] && !visited[i])
      DFSearch(i);
    }
```