SUSHANT
LBM19CS164

ADA - Test 1.

Warshall.C.

```
#include <stdio.h>
#define V 4
void printSolution(int reach[][V]);

void transitiveClosure(int graph[][V])
{
  int reach[V][V], i, j, k;
  for (i=0; i<V; i++)
    for (j=0; j<V; j++)
      reach[i][j] = graph[i][j];

for (k=0; k<V; k++)
{
  for (i=0; i<V; i++)
  {
    for (j=0; j<V; j++)
    {
      reach[i][j] = reach[i][j] ||
                    (reach[i][k] && reach
                    [k][j]);
    }
  }
}
print solution(reach);
}
```

```
void printSolution (int reach[V][V])
{
printf ("following matrix is transitive");
printf ("closure of the given graph \n");
for (int i=0; i<V; i++)
{
    for (int j= 0; j<V; j++)
    {
        if (i==j)
        printf("1 ");
        else
        printf("%3d", reach[i][j]);
    }
printf("\n");
}
}

int main()
{
int graph[V][V] = { {0,1,0,0},
                    {0,0,0,1},
                    {0,0,0,0},
                    {1,0,1,0}
                  };
transitiveClosure(graph);
return 0;
}
```

Modification

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

int A[20][20], visited[20],
  count = 0, n;
int seq[20], connected = 1,
 acyclic = 1;

void DFS();
void DFScore(int cur);
int main()
{
 int i, j;
 printf("\n Enter no. of vertices: ");
  scanf("%d", &n);
 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("It is not connected, acyclic");
if (connected && !acyclic)
printf(" connected & cyclic");
if (!connected && !acyclic)
printf(" Not connected & cyclic");

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

```
for (int i=1; i<=n; i++)
if (A[un](i) && !visited(i))
~~DFSearch(i);~~
DFSearch(i);
}
```