ASSIGNMENT NO:2 **DATE:09/02/2016**

PROGRAM TITLE:Traverse a Graph which is represented by Adjacency Matrix using Depth First Search.

PROGRAM ALGORITHM:

```
DFS(G,v)
{
      //input:Graph 'G' represented either by adjacency matrix, starting vertex
'v'
      //output:printing the vertices in DFS order
      push(S,v);//S is an empty Queue
      while S is not empty do
      {
            r=pop(S);
            if mark[r]==FALSE
            {
                  mark[u]=TRUE;
                  print(r);
                  for each adjacent vertex 'u' of 'r' do
                  {
                        push(S,u);
                  }
            }
      }
}
main()
{
      print(starting vertex);
      for each vertex u
            mark[u]=FALSE;
      DFS(G,v);
      for all unmarked vertices
            DFS(G,w);
}
```

PROGRAM CODE:

```
//C Code to Traverse a given graph using DFS using Adjacency Matrix
#include <stdio.h>
#include <stdlib.h>
#define MAX 50
int stack[MAX];
int top=-1;
void create_graph(int **graph,int n)
{
      int i,j,x;
      for(i=0;i<n;i++)
      {
            for(j=0;j<n;j++)
            {
                  graph[i][j]=0;
            }
      }
      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
                      graph[i][x]=1;
          }
     }
}
void print_graph(int **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");
}
int is_full()
{
     if(top==MAX-1)
          return 1;
     else
          return 0;
}
int is_empty()
{
     if(top==-1)
          return 1;
     else
          return 0;
}
void push(int n)
{
     if(is_full()==0)
     {
          top++;
          stack[top]=n;
     }
     else
          printf("\tThe Stack is Full.");
}
int pop()
{
     if(is_empty()==0)
     {
          return(stack[top--]);
```

```
	}
	else
		printf("\tThe Stack is Empty.");
	return -9999;
}
int dfs(int **graph,int n,int *mark,int v)
{
	int r,i;
	top=-1;
	push(v);
	while(is_empty()==0)
	{
		r=pop();
		if(r!=-9999)
		{
			if(mark[r]==0)
			{
				mark[r]=1;
				printf("\t%d",r);
				for(i=0;i<n;i++)
				{
					if(graph[r][i]==1)
					{
						push(i);
					}
				}
			}
		}
	}
	return 0;
}
int main()
{
	int n,i,v;
	printf("\n\tEnter the number of vertices::");
	scanf("%d",&n);
	int *mark=(int*)malloc(n*sizeof(int));
	int **graph= (int**)malloc(n*sizeof(int*));
	for(i=0;i<n;i++)
	{
		graph[i]=(int*)malloc(n*sizeof(int));
	}
	create_graph(graph,n);
	print_graph(graph,n);
	do
	{
		printf("\n\tEnter the starting vertex. Enter -99 to Quit::");
		scanf("%d",&v);
		for(i=0;i<n;i++)
			mark[i]=0;
		if(v!=-99)
		{
			printf("\tThe DFS traversal is::\n");
			dfs(graph,n,mark,v);
			printf("\n");
			for(i=0;i<n;i++)
			{
```

```
                        if(mark[i]==0)
                        {
                              dfs(graph,n,mark,i);
                              printf("\n");
                        }
                  }
            }
      }
      while(v!=-99);
      return 0;
}
```

OUTPUT:

```
      Enter the number of vertices::5
      Identify the adjoining vertices:
      Enter the adjoining vertices of 0.Enter -99 to stop::1 2 3
-99

      Enter the adjoining vertices of 1.Enter -99 to stop::0 4 -99

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

      Enter the adjoining vertices of 3.Enter -99 to stop::0 4 -99

      Enter the adjoining vertices of 4.Enter -99 to stop::1 2 3 -99

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

      Enter the starting vertex. Enter -99 to Quit::2
      The DFS traversal is::
      2     4     3     0     1

      Enter the starting vertex. Enter -99 to Quit::1
      The DFS traversal is::
      1     4     3     0     2

      Enter the starting vertex. Enter -99 to Quit::3
      The DFS traversal is::
      3     4     2     0     1

      Enter the starting vertex. Enter -99 to Quit::0
      The DFS traversal is::
      0     3     4     2     1

      Enter the starting vertex. Enter -99 to Quit::4
      The DFS traversal is::
      4     3     0     2     1

      Enter the starting vertex. Enter -99 to Quit::-99
```

DISCUSSION:

For Depth First Search, principle of Stack is used, therefore, we had to design some more functions to maintain and perform operations on the stack.