1. Graph traversal technique DFS (using stack)

#include<stdio.h>

#include<stdlib.h>

#define MAX 100

#define initial 1

#define visited 2

int n; /\* Number of nodes in the graph \*/

int adj[MAX][MAX]; /\*Adjacency Matrix\*/

int state[MAX]; /\*Can be initial or visited \*/

void DF\_Traversal();

void DFS(int v);

void create\_graph();

int stack[MAX];

int top = -1;

void push(int v);

int pop();

int isEmpty\_stack();

main()

{

create\_graph();

DF\_Traversal();

}/\*End of main()\*/

void DF\_Traversal()

{

int v;

for(v=0; v<n; v++)

state[v]=initial;

printf("\nEnter starting node for Depth First Search : ");

scanf("%d",&v);

DFS(v);

printf("\n");

}/\*End of DF\_Traversal( )\*/

void DFS(int v)

{

int i;

push(v);

while(!isEmpty\_stack())

{

v = pop();

if(state[v]==initial)

{

printf("%d ",v);

state[v]=visited;

}

for(i=n-1; i>=0; i--)

{

if(adj[v][i]==1 && state[i]==initial)

push(i);

}

}

}/\*End of DFS( )\*/

void push(int v)

{

if(top == (MAX-1))

{

printf("\nStack Overflow\n");

return;

}

top=top+1;

stack[top] = v;

}/\*End of push()\*/

int pop()

{

int v;

if(top == -1)

{

printf("\nStack Underflow\n");

exit(1);

}

else

{

v = stack[top];

top=top-1;

return v;

}

}/\*End of pop()\*/

int isEmpty\_stack( )

{

if(top == -1)

return 1;

else

return 0;

}/\*End if isEmpty\_stack()\*/

void create\_graph()

{

int i,max\_edges,origin,destin;

printf("\nEnter number of nodes : ");

scanf("%d",&n);

max\_edges=n\*(n-1);

for(i=1;i<=max\_edges;i++)

{

printf("\nEnter edge %d( -1 -1 to quit ) : ",i);

scanf("%d %d",&origin,&destin);

if( (origin == -1) && (destin == -1) )

break;

if( origin >= n || destin >= n || origin<0 || destin<0)

{

printf("\nInvalid edge!\n");

i--;

}

else

{

adj[origin][destin] = 1;

}

}

}

Output:

