**DAA PRACTICAL**

**Name:Parth Borse Roll No.: 07**

**Depth First search(DFS)**

#include<iostream>

#include<stdlib.h> // Include this header for using exit()

using namespace std;

#define MAX 100

#define initial 1

#define visited 2

int n;

int adj[MAX][MAX];

int state[MAX];

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();

int main() {

create\_graph();

DF\_traversal();

return 0; // This is where the main function should end

}

void DF\_traversal() {

int v;

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

state[v] = initial;

cout << "Enter starting node for Depth first search: ";

cin >> v;

DFS(v);

}

void DFS(int v) {

int i;

push(v);

while(!isEmpty\_stack()) {

v = pop();

if (state[v] == initial) {

cout << v << " "; // Print visited node

state[v] = visited;

}

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

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

push(i);

}

}

}

void push(int v) {

if (top == (MAX - 1)) {

cout << "\nStack overflow";

return;

}

top = top + 1;

stack[top] = v;

}

int pop() {

int v;

if (top == -1) {

cout << "\nStack underflow";

exit(1); // Exit the program if underflow occurs

}

else {

v = stack[top];

top = top - 1;

return v;

}

}

int isEmpty\_stack() {

if (top == -1)

return 1; // Return 1 if the stack is empty

else

return 0;

}

void create\_graph() {

int i, max\_edges, origin, destin;

cout << "Enter the number of nodes: ";

cin >> n;

max\_edges = n \* (n - 1); // Calculate maximum possible edges

for(i = 0; i < max\_edges; i++) {

cout << "Enter the edge " << i << " (or -1 -1 to stop): ";

cin >> origin >> destin;

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

break;

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

cout << "Invalid edge !\n";

i--; // Decrement i to re-enter edge

}

else {

adj[origin][destin] = 1; // Mark the edge

}

}

}

//Output:

