// DFS algorithm in C++

#include <iostream>

#include<vector>

#include<stack>

using namespace std;

class Graph {

int numVertices;

vector<int>\* adjLists;

bool\* visited;

public:

Graph(int vertices);

void addEdge(int src, int dest);

void DFS(int startVertex);

};

// Create a graph with given vertices,

// and maintain an adjacency list

Graph::Graph(int vertices) {

numVertices = vertices;

adjLists = new vector<int>[vertices];

}

// Add edges to the graph

void Graph::addEdge(int src, int dest) {

adjLists[src].push\_back(dest);

adjLists[dest].push\_back(src);

}

// DFS algorithm

//==========

void Graph:: DFS(int startVertex)

{

visited = new bool[numVertices];

for (int i = 0; i < numVertices; i++)

visited[i] = false;

stack<int>stk;//stack in STL

stk.push(startVertex);

visited[startVertex]=true;

while(!stk.empty())

{

int u=stk.top();

cout<<u<<" ";

stk.pop();

//loop for traverse

for(int i=0;i<adjLists[u].size();i++){

if(!visited[adjLists[u][i]]){

stk.push(adjLists[u][i]);

visited[adjLists[u][i]]=true;

}

}

}

}

int main() {

Graph g(5);

g.addEdge(0, 4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 3);

g.addEdge(3, 3);

g.DFS(2);

return 0;

}