**Name:** Rohan Arun Nalawade

**PRN:** 22310407

**Roll No:** 231012

**SY IT A**

**Assignment 2**

**BFS & DFS**

**Aim:** Create a graph and perform graph traversal using BFS and DFS

**Code:**

#include<iostream>

#include<queue>

#include<vector>

using namespace std;

class node {

public:

int name;

node\* next;

node(int d) {

name = d;

next = NULL;

}

};

class Graph {

vector<node \*> gr;

vector<bool> visited;

int size;

public:

Graph() {

size = 0;

}

void InsertNode()

{

gr.push\_back(new node(size));

visited.push\_back(false);

cout << "Node " << size << " inserted" << endl;

size++;

}

void InsertEdge(int x, int y) {

if (x < size && y < size) {

node\* temp = gr[x];

while(temp->next != NULL) {

temp = temp->next;

}

temp->next = new node(y);

if (x != y) {

temp = gr[y];

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = new node(x);

}

}

else {

cout << "One or both nodes missing" << endl;

}

}

void dfs(int v = 0) {

node\* w;

visited[v] = true;

cout << gr[v]->name << " ";

for(w=gr[v]; w!=NULL; w=w->next)

{

if(!visited[w->name])

{

dfs(w->name);

}

}

}

void clear\_visited()

{

 for (int i = 0; i < 20; i++)

 {

 visited[i] = false;

 }

}

void bfs(int v=0)

{

 clear\_visited();

 node \* w;

 queue<int> q;

 cout<<v<<" ";

 visited[v]=true;

 q.push(v);

 while(!q.empty())

 {

  v = q.front();

  q.pop();

  for(w=gr[v]; w!=NULL ; w=w->next)

  {

   if(!visited[w->name])

   {

    cout<<w->name<<" ";

    q.push(w->name);

    visited[w->name] = true;

   }

  }

 }

}

};

int main() {

int n, choice = 1, x, y;

Graph g;

cout << "Enter how many nodes you want to enter: ";

cin >> n;

for (int i = 0; i < n; i++) {

g.InsertNode();

}

do {

cout << "Enter the nodes to insert edge (x y): ";

cin >> x >> y;

if (x >= 0 && x < n && y >= 0 && y < n) {

g.InsertEdge(x, y);

}

else {

cout << "Invalid node indices. Please try again." <<

endl;

}

cout << "Do you want to insert another edge? Yes1 or No-2: ";

cin >> choice;

} while (choice == 1);

cout << "\nDepth First Search (DFS) starting from node 0:\n";

g.dfs();

cout << endl;

cout << "\nBreadth First Search (BFS) starting from node 0:\n";

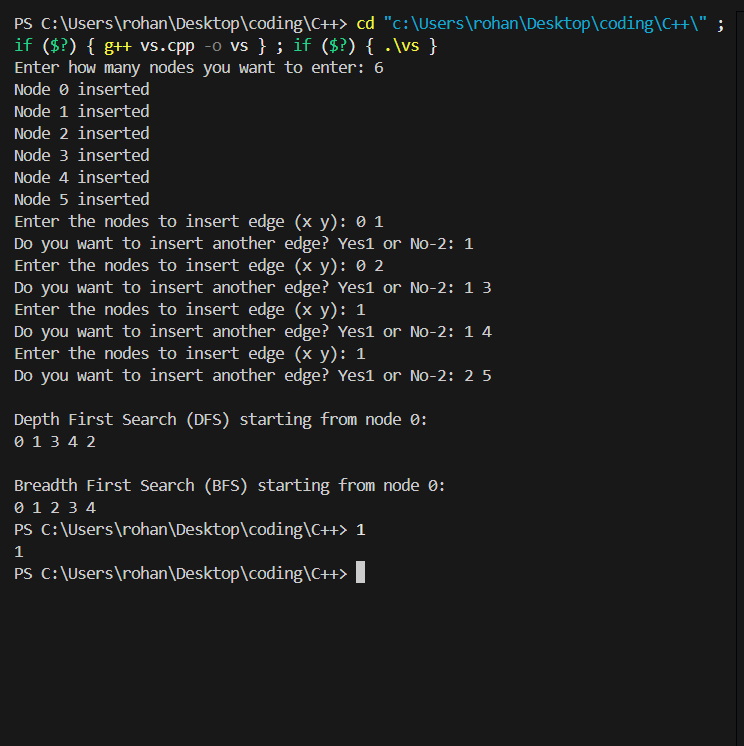
g.bfs();

cout << endl;

return 0;

}

**Output:**

****