*main function:*

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

// Name : 21118\_DSA\_A06.cpp

// Author : Shubham (Roll No: 21118)

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

**#include** <iostream>

**#include** "Graph.h"

**using** **namespace** std;

**int** **main**() {

**while** (1) {

cout << "\n\n\*\*NOTE: 1-based INDEXING\*\*\n";

**int** n, m, u, v, opt, src;

cout << "Enter number of vertices and edges:\n";

cin >> n >> m;

Graph g(n, m);

cout << "Enter edges (input format: vertex1 <space> vertex2)\n";

**for** (**int** i = 0; i < m; i++) {

cin >> u >> v;

g.addEdge(u, v);

}

cout << "Printing Graph:\n";

g.PrintGraph();

cout << "Enter\n";

cout << "\t1 for Breadth-first-traversal.\n";

cout << "\t2 for Depth-first-traversal.\n";

cout << "\t0 to exit.\n";

cout << ": ";

cin >> opt;

**if** (opt != 0) {

cout << "Enter Source vertex: ";

cin >> src;

}

**switch** (opt) {

**case** 0: {

**break**;

}

**case** 1: {

cout << "Breadth-first-traversal: ";

g.bfs(src);

cout << **endl**;

**break**;

}

**case** 2: {

cout << "Depth-first-traversal: ";

g.dfs(src);

cout << **endl**;

**break**;

}

**default**:

cout << "INVALID CHOICE.\n";

}

}

**return** 0;

}

*Graph.h header file (contains graph class-declaration):*

/\*

\* Graph.h

\*

\* Created on: 18-May-2021

\* Author: Shubham

\*/

**#ifndef** GRAPH\_H\_

**#define** GRAPH\_H\_

**#include** <iostream>

**using** **namespace** std;

**class** Node {

**private**:

**int** data;

Node \*next;

**public**:

**Node**(**int** x = 0);

**friend** **class** Graph;

};

**class** Graph {

**private**:

**int** vertices, edges;

Node \*\*adj\_list;

Node\* **Insert**(Node \*&head, **int** x);

**public**:

**Graph**(**int** n, **int** m);

**void** **addEdge**(**int** u, **int** v);

**void** **PrintGraph**();

**void** **bfs**(**int** src);

**void** **dfs\_ut**(**int** u, **bool** vis[]);

**void** **dfs**(**int** src);

**void** **dfs\_it**(**int** src);

};

**#endif** /\* GRAPH\_H\_ \*/

*Graph.cpp file (contains Graph class implementations):*

/\*

\* Graph.cpp

\*

\* Created on: 18-May-2021

\* Author: Shubham

\*/

**#include** <iostream>

**#include** "STACK.h"

**#include** "QUEUE.h"

**#include** "Graph.h"

**using** **namespace** std;

// Node class

**Node::Node**(**int** x) {

data = x;

next = NULL;

}

// Graph Class

**Graph::Graph**(**int** n, **int** m) {

vertices = n, edges = m;

adj\_list = **new** Node\*[n + 1];

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

adj\_list[i] = NULL;

}

Node\* **Graph::Insert**(Node \*&head, **int** x) {

Node \*newNode = **new** Node(x);

**if** (head == NULL)

head = newNode;

**else** {

Node \*tmp = head;

**while** (tmp->next)

tmp = tmp->next;

tmp->next = newNode;

}

**return** head;

}

**void** **Graph::addEdge**(**int** u, **int** v) {

adj\_list[u] = Insert(adj\_list[u], v); // inserting at the end of list

adj\_list[v] = Insert(adj\_list[v], u);

}

**void** **Graph::PrintGraph**() {

**for** (**int** i = 1; i <= vertices; i++) {

cout << i << " -> ";

**for** (Node \*head = adj\_list[i]; head; head = head->next)

cout << head->data << " ";

cout << **endl**;

}

}

**void** **Graph::bfs**(**int** src) {

QUEUE<**int**> qu(100);

**bool** vis[vertices + 1] = { 0 };

qu.Push(src);

vis[src] = 1;

**while** (!qu.Empty()) {

**int** u = qu.Front();

qu.Pop();

cout << u << " ";

**for** (Node \*head = adj\_list[u]; head; head = head->next)

**if** (!vis[head->data]) {

vis[head->data] = 1;

qu.Push(head->data);

}

}

}

**void** **Graph::dfs\_ut**(**int** u, **bool** vis[]) {

vis[u] = 1;

cout << u << " ";

**for** (Node \*head = adj\_list[u]; head; head = head->next)

**if** (!vis[head->data])

dfs\_ut(head->data, vis);

}

**void** **Graph::dfs**(**int** src) {

**bool** vis[vertices + 1] = { 0 };

dfs\_ut(src, vis);

}

**void** **Graph::dfs\_it**(**int** src) {

STACK<**int**> stk(100);

**bool** vis[vertices+1]={0};

stk.Push(src);

vis[src]=1;

**while** (!stk.Empty()) {

**int** u = stk.Top();

stk.Pop();

cout << u << " ";

**for** (Node\* head = adj\_list[u]; head; head=head->next)

**if** (!vis[head->data]) {

vis[head->data] = 1;

stk.Push(head->data);

}

}

}

*STACK.h header file:*

/\*

\* STACK.h

\*

\* Created on: 24-May-2021

\* Author: Shubham

\*/

**#ifndef** STACK\_H\_

**#define** STACK\_H\_

**template** <**typename** **T**>

**class** STACK {

**private**:

**T**\* a;

**int** top, size;

**public**:

**STACK**(**int** size) {

top=-1, **this**->size = size;

a = **new** **T**[size];

}

**bool** **Empty**() {

**return** (top==-1);

}

**void** **Push**(**T** x) {

top++;

**if** (top < size)

a[top] = x;

**else** top--;

}

**void** **Pop**() {

**if** (!Empty())

top--;

}

**T** **Top**() {

**if** (!Empty())

**return** a[top];

**else** **return** -1;

}

**~STACK**() {

**delete**[] a;

}

};

**#endif** /\* STACK\_H\_ \*/

*QUEUE.h header file:*

/\*

\* QUEUE.h

\*

\* Created on: 18-May-2021

\* Author: Shubham

\*/

**#ifndef** QUEUE\_H\_

**#define** QUEUE\_H\_

**template** <**typename** **T**>

**class** QUEUE {

**private**:

**T**\* a;

**int** front, size, rear;

**public**:

**QUEUE**(**int** size) {

front = 0, rear = 0;

**this**->size = size;

a = **new** **T**[size];

}

**bool** **Empty**() {

**return** front == rear;

}

**bool** **isFull**() {

**int** temp = (rear + 1) % size;

**return** (temp == front);

};

**void** **Push**(**T** x) {

**if** (!isFull()) {

rear = (rear + 1) % size;

a[rear] = x;

}

}

**void** **Pop**() {

**if** (Empty()) **return**;

front = (front + 1) % size;

}

**T** **Front**() {

**return** a[front+1];

}

**~QUEUE**() {

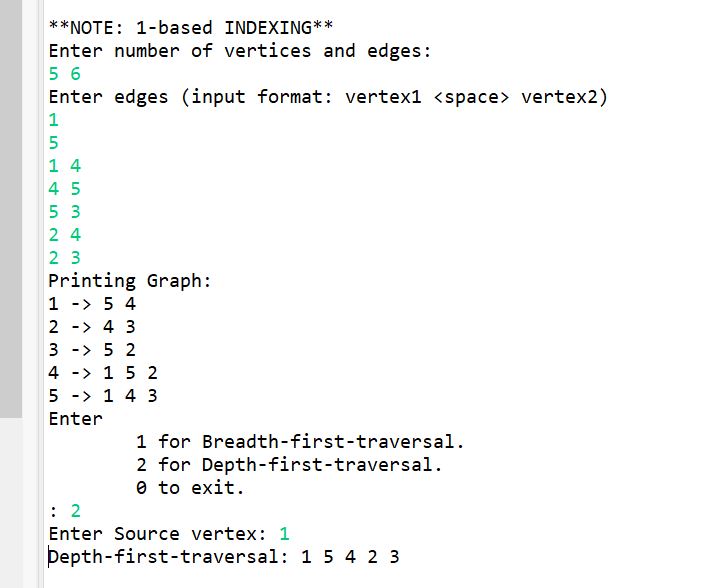
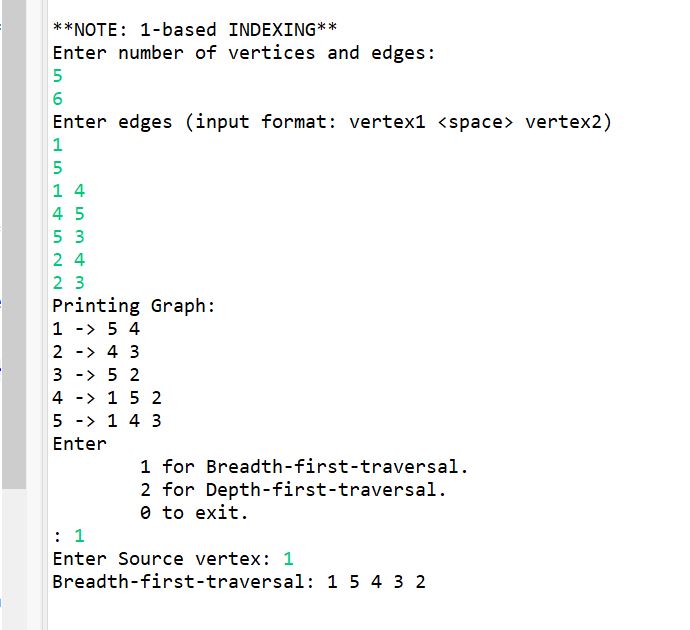
**delete**[] a;

}

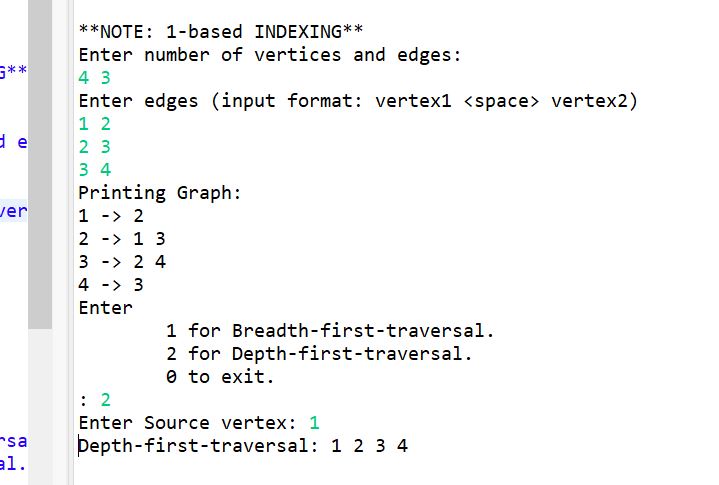
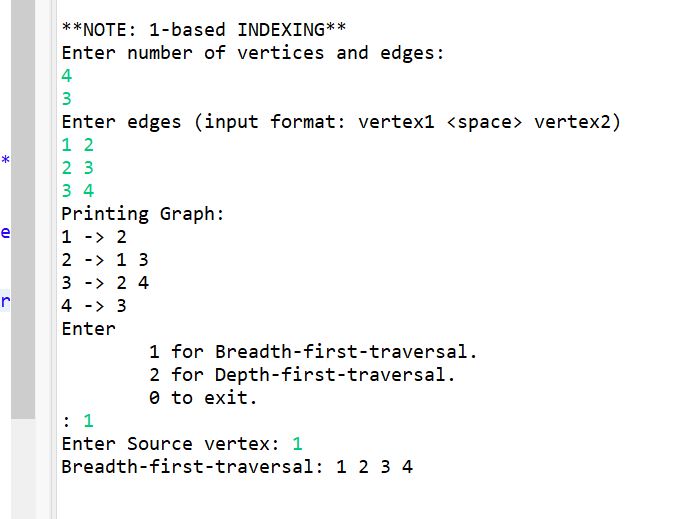
};

**#endif** /\* QUEUE\_H\_ \*/

TESTCASE 1:



TESTCASE 2:



TESTCASE 3:

