***TASK 1:***

#include<iostream>

#include <vector>

#include <string>

#include<iomanip>

using namespace std;

class Graph {

public:

int v;//vertices

int e;//edges

int\*\* adj;//matrix

void print\_Matrix(int n);

Graph(int v, int e);

void S(int start, vector<bool>& visited);

void addEdge(int start, int e);

void add\_Edge(char a, char b);

};

Graph::Graph(int v, int e)

{

this->v = v;

this->e = e;

adj = new int\* [v];

for (int row = 0; row < v; row++)

{

adj[row] = new int[v];

for (int column = 0; column < v; column++)

{

adj[row][column] = 0;

}

}

}

void Graph::print\_Matrix(int n)

{

int i, j;

cout << "\n\n" << setw(3) << "";

cout << " ";

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

{

cout <<setw(2) << char(65 + j);

cout << "|";

cout << " ";

}

cout << "\n\n";

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

{

cout << setw(3) << char(65 + i) << "|";

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

{

cout << setw(4) << adj[i][j];

}

cout << "\n\n";

}

}

void Graph::S(int start, vector<bool>& visited)

{

cout << char(65 + start) << " ";

visited[start] = true;

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

if (adj[start][i] == 1 && (!visited[i])) {

S(i, visited);

}

}

}

void Graph::addEdge(int start, int e)

{

adj[start][e] = 1;

adj[e][start] = 1;

}

void Graph::add\_Edge(char a, char b)

{

int first, second;

first = int(a) - 65;

second = int(b) - 65;

addEdge(first, second);

}

int main()

{

int a, b;

char A, B;

int v = 6, e = 7;

Graph Obj(v, e);

Obj.add\_Edge('A', 'B');

Obj.add\_Edge('A', 'C');

Obj.add\_Edge('A', 'D');

Obj.add\_Edge('B', 'C');

Obj.add\_Edge('B', 'E');

Obj.add\_Edge('D', 'E');

Obj.add\_Edge('D', 'F');

Obj.print\_Matrix(5);

cout << endl;

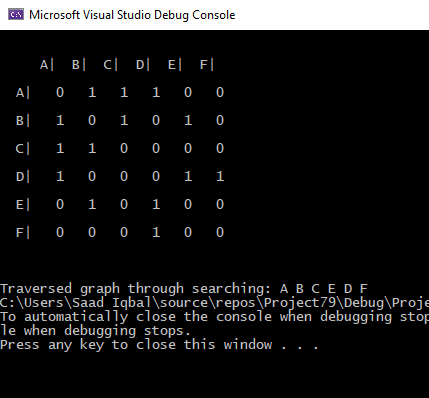
cout << endl;

vector<bool> visited(v, false);

cout << "Traversed graph through searching: ";

Obj.S(0, visited);

}

******

***TASK 2:***

#include<iostream>

#include <vector>

#include <string>

#include<iomanip>

using namespace std;

class Graph {

public:

int v;//vertices

int e;//edges

int\*\* adj;//matrix

void print\_Matrix(int n);

Graph(int v, int e);

void BFS(int start);

void addEdge(int start, int e);

void add\_Edge(char a, char b);

};

Graph::Graph(int v, int e)

{

this->v = v;

this->e = e;

adj = new int\* [v];

for (int row = 0; row < v; row++)

{

adj[row] = new int[v];

for (int column = 0; column < v; column++)

{

adj[row][column] = 0;

}

}

}

void Graph::print\_Matrix(int n)

{

int i, j;

cout << "\n\n" << setw(3) << "";

cout << " ";

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

{

cout << setw(2) << char(65 + j);

cout << "|";

cout << " ";

}

cout << "\n\n";

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

{

cout << setw(3) << char(65 + i) << "|";

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

{

cout << setw(4) << adj[i][j];

}

cout << "\n\n";

}

}

void Graph::BFS(int start)

{

vector<bool> visited(v, false);

vector<int> q;

q.push\_back(start);

visited[start] = true;

int vis;

while (!q.empty())

{

vis = q[0];

cout << char(65 + vis) << " ";

q.erase(q.begin());

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

if (adj[vis][i] == 1 && (!visited[i])) {

q.push\_back(i);

visited[i] = true;

}

}

}

}

void Graph::addEdge(int start, int e)

{

adj[start][e] = 1;

adj[e][start] = 1;

}

void Graph::add\_Edge(char a, char b)

{

int first, second;

first = int(a) - 65;

second = int(b) - 65;

addEdge(first, second);

}

int main()

{

int a, b;

char A, B;

int v = 6, e = 7;

Graph Obj(v, e);

Obj.add\_Edge('A', 'B');

Obj.add\_Edge('A', 'C');

Obj.add\_Edge('A', 'D');

Obj.add\_Edge('B', 'C');

Obj.add\_Edge('B', 'E');

Obj.add\_Edge('D', 'E');

Obj.add\_Edge('D', 'F');

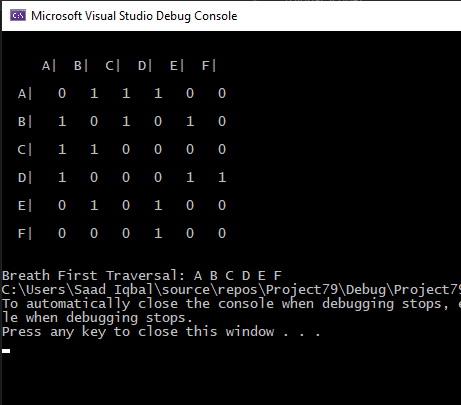
Obj.print\_Matrix(5);

cout << endl;

cout << "Breath First Traversal: ";

Obj.BFS(0);

}



***TASK 3:***

#include<iostream>

#include <vector>

#include <string>

#include<iomanip>

using namespace std;

class Graph {

public:

int v;//vertices

int e;//edges

int\*\* adj;//matrix

void print\_Matrix(int n);

Graph(int v, int e);

void DFS(int start, vector<bool>& visited);

void addEdge(int start, int e);

void add\_Edge(char a, char b);

};

Graph::Graph(int v, int e)

{

this->v = v;

this->e = e;

adj = new int\* [v];

for (int row = 0; row < v; row++)

{

adj[row] = new int[v];

for (int column = 0; column < v; column++)

{

adj[row][column] = 0;

}

}

}

void Graph::print\_Matrix(int n)

{

int i, j;

cout << "\n\n" << setw(3) << "";

cout << " ";

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

{

cout <<setw(2) << char(65 + j);

cout << "|";

cout << " ";

}

cout << "\n\n";

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

{

cout << setw(3) << char(65 + i) << "|";

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

{

cout << setw(4) << adj[i][j];

}

cout << "\n\n";

}

}

void Graph::DFS(int start, vector<bool>& visited)

{

cout << char(65 + start) << " ";

visited[start] = true;

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

if (adj[start][i] == 1 && (!visited[i])) {

DFS(i, visited);

}

}

}

void Graph::addEdge(int start, int e)

{

adj[start][e] = 1;

adj[e][start] = 1;

}

void Graph::add\_Edge(char a, char b)

{

int first, second;

first = int(a) - 65;

second = int(b) - 65;

addEdge(first, second);

}

int main()

{

int a, b;

char A, B;

int v = 6, e = 7;

Graph Obj(v, e);

Obj.add\_Edge('A', 'B');

Obj.add\_Edge('A', 'C');

Obj.add\_Edge('A', 'D');

Obj.add\_Edge('B', 'C');

Obj.add\_Edge('B', 'E');

Obj.add\_Edge('D', 'E');

Obj.add\_Edge('D', 'F');

Obj.print\_Matrix(5);

cout << endl;

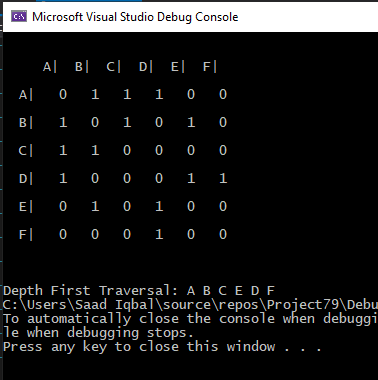
cout << endl;

vector<bool> visited(v, false);

cout << "Depth First Traversal: ";

Obj.DFS(0, visited);

}

******

***TASK 4:***

#include<iostream>

#include <vector>

#include <string>

#include<iomanip>

using namespace std;

class Graph {

public:

int v;//vertices

int e;//edges

int\*\* adj;//matrix

void print\_Matrix(int n);

Graph(int v, int e);

void BFS(int start);

void DFS(int start, vector<bool>& visited);

void addEdge(int start, int e);

void add\_Edge(char a, char b);

};

Graph::Graph(int v, int e)

{

this->v = v;

this->e = e;

adj = new int\* [v];

for (int row = 0; row < v; row++)

{

adj[row] = new int[v];

for (int column = 0; column < v; column++)

{

adj[row][column] = 0;

}

}

}

void Graph::print\_Matrix(int n)

{

int i, j;

cout << "\n\n" << setw(3) << "";

cout << " ";

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

{

cout <<setw(2) << char(65 + j);

cout << "|";

cout << " ";

}

cout << "\n\n";

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

{

cout << setw(3) << char(65 + i) << "|";

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

{

cout << setw(4) << adj[i][j];

}

cout << "\n\n";

}

}

void Graph::BFS(int start)

{

vector<bool> visited(v, false);

vector<int> q;

q.push\_back(start);

visited[start] = true;

int vis;

while (!q.empty())

{

vis = q[0];

cout << char(65 + vis) << " ";

q.erase(q.begin());

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

if (adj[vis][i] == 1 && (!visited[i])) {

q.push\_back(i);

visited[i] = true;

}

}

}

}

void Graph::DFS(int start, vector<bool>& visited)

{

cout << char(65 + start) << " ";

visited[start] = true;

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

if (adj[start][i] == 1 && (!visited[i])) {

DFS(i, visited);

}

}

}

void Graph::addEdge(int start, int e)

{

adj[start][e] = 1;

adj[e][start] = 1;

}

void Graph::add\_Edge(char a, char b)

{

int first, second;

first = int(a) - 65;

second = int(b) - 65;

addEdge(first, second);

}

int main()

{

int a, b;

char A, B;

int v = 13, e = 12;

Graph Obj(v, e);

Obj.add\_Edge('A', 'B');

Obj.add\_Edge('A', 'C');

Obj.add\_Edge('B', 'D');

Obj.add\_Edge('B', 'E');

Obj.add\_Edge('D', 'G');

Obj.add\_Edge('D', 'H');

Obj.add\_Edge('G', 'L');

Obj.add\_Edge('E', 'I');

Obj.add\_Edge('I', 'M');

Obj.add\_Edge('C', 'F');

Obj.add\_Edge('F', 'J');

Obj.print\_Matrix(10);

cout << endl;

cout << "Breath First Traversal From A to G: ";

Obj.BFS(0);

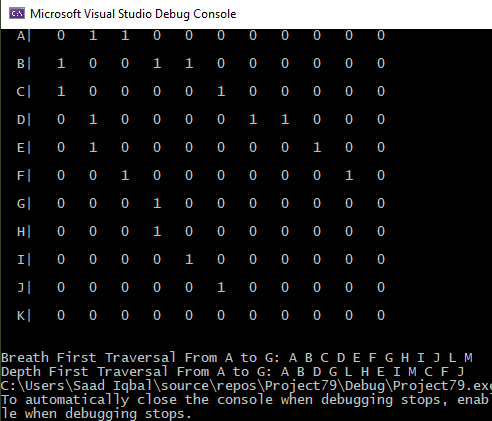
cout << endl;

vector<bool> visited(v, false);

cout << "Depth First Traversal From A to G: ";

Obj.DFS(0, visited);

}

******