//Assignment no 6

#include <iostream>

using namespace std;

class graph

{

public:

int vcnt,rcnt;

int a[20][20];

int v[20];

graph()

{

vcnt=0;

rcnt=0;

}

void admat();

void dis\_admat();

void bfs();

void dfs();

int search(int key);

};

void graph::admat()

{

cout<<"Enter total no. of vertices:";

cin>>vcnt;

for(int i=1;i<=vcnt;i++)

{

for(int j=1;j<=vcnt;j++)

{

cout<<"Edge from "<<i<<" to "<<j<<":";

cin>>a[i][j];

}

}

}

void graph::dis\_admat()

{

for(int i=1;i<=vcnt;i++)

{

for(int j=1;j<=vcnt;j++)

{

cout<<a[i][j]<<" ";

}

cout<<endl;

}

}

class queue

{

public:

int q[20];

int front,rear;

queue()

{

front=0;

rear=0;

}

void insert(int key);

int del();

};

void queue::insert(int key)

{

if(rear<20)

{

q[rear++]=key;

}

else

{

cout<<"Queue is full."<<endl;

}

}

int queue::del()

{

if(front!=rear)

{

return(q[front++]);

}

else

{

cout<<"Queue is empty."<<endl;\

return -1;

}

}

class stack

{

public:

int s[20];

int top;

stack()

{

top=-1;

}

void push(int key);

int pop();

};

void stack::push(int key)

{

if(top<=20)

{

s[++top]=key;

}

else

{

cout<<"Stack is full."<<endl;

}

}

int stack::pop()

{

if(top!=-1)

{

return s[top--];

}

else

{

cout<<"Stack is empty."<<endl;

return -1;

}

}

int graph::search(int key)

{

//search val in visited array. if val is already present return 1 else return 0

int i;

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

{

if(v[i]==key)

{

return (1);

}

}

if(i==rcnt)

{

return (0);

}

return -1;

}

void graph::bfs()

{

int sv,curr; //sv= start vertex

cout<<"Enter start vertex:";

cin>>sv;

v[rcnt++]=sv;

queue q;

q.insert(sv);

for(int i=1;i<=vcnt;i++)

{

curr=q.del(); //curr will store the value at the front of queue

for(int c=1;c<=vcnt;c++)

{

if(a[curr][c]==1) //check for the adjacent vertice of curr

{

q.insert(c); //insert adjacent vertices of curr in the queue

if(search(c)==0)

{

v[rcnt++]=c;

}

}

}

}

cout<<"BFS:";

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

{

cout<<v[i]<<" ";

}

cout<<endl;

}

void graph::dfs()

{

int sv,curr; //sv= start vertex

cout<<"Enter start vertex:";

cin>>sv;

v[rcnt++]=sv;

stack s;

s.push(sv);

for(int i=1;i<=vcnt;i++)

{

curr=s.pop(); //curr will store the value at the front of queue

for(int c=1;c<=vcnt;c++)

{

if(a[curr][c]==1) //check for the adjacent vertice of curr

{

s.push(c); //insert adjacent vertices of curr in the queue

if(search(c)==0)

{

v[rcnt++]=c;

}

}

}

}

cout<<"DFS:";

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

{

cout<<v[i]<<" ";

}

cout<<endl;

}

int main()

{

graph g;

int ch;

do{

cout<<"---MENU---\n";

cout<<"1.Insert Matrix\n";

cout<<"2.Display Matrix\n";

cout<<"3.BFS\n";

cout<<"4.DFS\n";

cout<<"5.Exit\n";

cout<<"Enter your choice\n";

cin>>ch;

switch(ch){

case 1:

{

g.admat();

break;

}

case 2:

{

g.dis\_admat();

break;

}

case 3:

{

g.bfs();

break;

}

case 4:

{

g.dfs();

break;

}

case 5:

{

cout<<"End of the program";

break;

}

default:

cout<<"Invalid choice!";

}

}while(ch!=5);

return 0;

}

output:

gescoe@gescoe-OptiPlex-3010:~/Desktop/SE-A-55$ g++ graph.cpp

gescoe@gescoe-OptiPlex-3010:~/Desktop/SE-A-55$ ./a.out

---MENU---

1. Insert Matrix
2. Display Matrix
3. BFS
4. DFS
5. Exit

Enter your choice

1

Enter total no. of vertices:4

Edge from 1 to 1:1

Edge from 1 to 2:2

Edge from 1 to 3:3

Edge from 1 to 4:4

Edge from 2 to 1:5

Edge from 2 to 2:6

Edge from 2 to 3:7

Edge from 2 to 4:8

Edge from 3 to 1:9

Edge from 3 to 2:10

Edge from 3 to 3:11

Edge from 3 to 4:12

Edge from 4 to 1:13

Edge from 4 to 2:14

Edge from 4 to 3:15

Edge from 4 to 4:16

---MENU---

1. Insert Matrix
2. Display Matrix
3. BFS
4. DFS
5. Exit

Enter your choice

2

1. 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

---MENU---

1. Insert Matrix
2. Display Matrix
3. BFS
4. DFS
5. Exit

Enter your choice

3

Enter start vertex:1

BFS:1

---MENU---

1. Insert Matrix
2. Display Matrix
3. BFS
4. DFS
5. Exit

Enter your choice

2

1. 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

---MENU---

1. Insert Matrix
2. Display Matrix
3. BFS
4. DFS
5. Exit

Enter your choice

4

Enter start vertex:2

Stack is empty.

Stack is empty.

Stack is empty.

DFS:1 2

---MENU---

1. Insert Matrix
2. Display Matrix
3. BFS
4. DFS
5. Exit

Enter your choice

2

1. 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

---MENU---

1. Insert Matrix
2. Display Matrix
3. BFS
4. DFS
5. Exit

Enter your choice

5

End of the programgescoe@gescoe-OptiPlex-3010:~/Desktop/SE-A-55$