#include<iostream>

#include<iomanip>

using namespace std;

#define MAX 100

class stack

{

public:

int st[20];

int data,top;

public:

stack()

{

top = -1;

}

int isEmpty()

{

if(top==-1)

return 1;

else

return 0;

}

void push(int new\_val)

{

top++;

st[top] = new\_val;

}

int pop()

{

int new\_val;

new\_val = st[top];

top--;

return(new\_val);

}

};

class queue

{

int que[100];

int data,rear,front;

public:

queue()

{

rear = front = -1;

}

int isEmpty()

{

if(rear==front)

return 1;

else

return 0;

}

int isFull()

{

if(rear==20)

return 1;

else

return 0;

}

void add(int new\_val)

{

if(isFull())

cout<<"\nQueue Overflow";

else

{

rear++;

que[rear] = new\_val;

}

}

int del()

{

int new\_val;

if(isEmpty())

{

cout<<"\nQueue is Empty";

}

else

{

front++;

new\_val = que[front];

return(new\_val);

}

}

};

class graph

{

private:

int i, j, v1, v2, tmp\_weight, starting\_vertex, mat[20][20];

int visited[20][20];

public:

int v, visited1[100];

void create();

void display();

void bfs(int i);

void dfs(int i);

};

void graph::create()

{

int ch;

cout<<"\nEnter number of vertices : ";

cin>>v;

for(i = 1; i <= v; i++)

{

for(j = i; j <= v; j++)

{

if(i != j)

{

cout<<"\nEnter 1 if the vertex "<<i<<" is connected to "<<j<<"otherwise 0: ";

cin>>ch;

if(ch==1)

{

mat[i][j] = 1;

}

else if(ch==0)

{

mat[i][j] = 0;

}

mat[j][i] = mat[i][j];

}

else

{

mat[i][j] = 0;

}

}

}

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

visited1[i]=0;

cout<<"\nGraph Created Successfully!"<<endl;

}

void graph::display()

{

cout<<"\n\n"<<setw(4)<<"";

for(i = 1; i <= v; i++)

cout<<setw(3)<<"("<<i<<")";

cout<<"\n\n";

for(i = 1; i <= v; i++)

{

cout<<setw(3)<<"("<<i<<")";

for(j = 1; j <= v; j++)

{

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

}

cout<<"\n\n";

}

}

void graph :: bfs(int i)

{

int k;

queue Q;

visited1[i] = 1;

Q.add(i);

while(!Q.isEmpty())

{

j = Q.del();

cout<<j;

for(k = 1; k <= v; k++)

{

if(mat[j,k] && !visited1[k])

{

Q.add(k);

visited1[k] = 1;

}

}

}

}

void graph::dfs(int i)

{

stack S;

S.push(i);

while(!S.isEmpty())

{

i = S.pop();

if(visited1[i]!=1)

{

cout<<i;

visited1[i] = 1;

}

for(int k=1; k<=v; k++)

{

if(mat[i][k]==1 && visited1[k]==0)

{

S.push(k);

}

}

}

}

int main()

{

graph obj;

int choice, start\_vertex;

while(1)

{

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

cout<<"\n1. Create a Graph\n2. Breadth-First Search Traversal\n3. Depth-First Search Traversal\n4. Display Graph\n5. Exit";

cout<<"\nEnter your choice : ";

cin>>choice;

switch(choice)

{

case 1: obj.create();

break;

case 2: cout<<"\nBreadth-First Search Traversal : ";

for(int k = 1; k <= obj.v; k++)

obj.visited1[k] = 0;

obj.bfs(1);

cout<<"\n";

break;

case 3: cout<<"\nDepth-First Search Traversal : ";

for(int k = 1; k <= obj.v; k++)

obj.visited1[k] = 0;

obj.dfs(1);

cout<<"\n";

break;

case 4: obj.display();

break;

case 5: cout<<"\nProgram Exited!";

exit(0);

break;

default: cout<<"\nInvalid Input";

}

}

return 0;

}

OUTPUT:

---------------------------------------------------------

1. Create a Graph

2. Breadth-First Search Traversal

3. Depth-First Search Traversal

4. Display Graph

5. Exit

Enter your choice : 1

Enter number of vertices : 3

Enter 1 if the vertex 1 is connected to 2otherwise 0: 1

Enter 1 if the vertex 1 is connected to 3otherwise 0: 1

Enter 1 if the vertex 2 is connected to 3otherwise 0: 0

Graph Created Successfully!

---------------------------------------------------------

1. Create a Graph

2. Breadth-First Search Traversal

3. Depth-First Search Traversal

4. Display Graph

5. Exit

Enter your choice : 2

Breadth-First Search Traversal : 123

---------------------------------------------------------

1. Create a Graph

2. Breadth-First Search Traversal

3. Depth-First Search Traversal

4. Display Graph

5. Exit

Enter your choice : 3

Depth-First Search Traversal : 132

---------------------------------------------------------

1. Create a Graph

2. Breadth-First Search Traversal

3. Depth-First Search Traversal

4. Display Graph

5. Exit

Enter your choice : 4

(1) (2) (3)

(1) 0 1 1

(2) 1 0 0

(3) 1 0 0

---------------------------------------------------------