**Assignment No:-**

**Assignment Name**:- **Write a program to implement breadth first and depth first search.**

**Name:- WARKE PURVA DILIP.**

**Roll No:- 136.**

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

#include<iostream.h>

#include<conio.h>

#include<stdlib.h>

int A[][9]=

{

{0,0,0,0,0,0,0,0,0},

{0,0,1,1,0,0,0,0,0},

{0,1,0,0,1,1,0,0,0},

{0,1,0,0,0,0,1,1,0},

{0,0,1,0,0,0,0,0,1},

{0,0,1,0,0,0,0,0,1},

{0,0,0,1,0,0,0,0,1},

{0,0,0,1,0,0,0,0,1},

{0,0,0,0,1,1,1,1,0}

};

class GRAPH

{

private:

//int n,A[10][10];

int n;

int VISITED[10];

public:

GRAPH(int);

void READ\_GRAPH();

void SHOW\_GRAPH();

void DFS\_R(int);

void DFS\_S(int);

void BFS\_Q(int);

};

GRAPH::GRAPH(int par)

{

n=par;

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

VISITED[i]=0;

}

void GRAPH::READ\_GRAPH()

{

/\* cout<<"\nEnter Adj matrix : ";

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

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

cin>>A[i][j];

\*/

}

void GRAPH::SHOW\_GRAPH()

{

cout<<endl;

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

{

cout<<endl;

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

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

}

}

void GRAPH::DFS\_R(int v)

{

VISITED[v]=1;

cout<<v<<" ";

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

{

if(A[v][i]==1 && VISITED[i]==0)

DFS\_R(i);

}

}

void GRAPH::DFS\_S(int v)

{

int STK[10],top=0;

int u=v;

VISITED[v]=1;

do

{

cout<<u<<" ";

for(int w=1;w<=n;w++)

{ // adj && not visited

if(A[u][w]==1 && VISITED[w]==0)

{

top=top+1;

STK[top]=w;

VISITED[w]=1;

}

}

if(top==0)

break;

else

{

u=STK[top];

top=top-1;

}

} while(1);

}

void GRAPH::BFS\_Q(int v)

{

int QUE[10],rear=0,front=0;

int u=v;

VISITED[v]=1;

do

{

cout<<u<<" ";

for(int w=1;w<=n;w++)

{ // adj && not visited

if(A[u][w]==1 && VISITED[w]==0)

{

rear=rear+1;

QUE[rear]=w;

if(front==0)

front=1;

VISITED[w]=1;

}

}

if(front==0) // Q empty

break;

else

{

u=QUE[front];

if(front==rear)

front=0;

else

front=front+1;

}

} while(1);

}

void main()

{

int n;

clrscr();

cout<<"\n Enter no of nodes : ";

cin>>n;

GRAPH obj(n);

obj.READ\_GRAPH();

obj.SHOW\_GRAPH();

cout<<"\n\n BFS is : ";

obj.BFS\_Q(4);

//cout<<"\n\n DFS is : ";

//obj.DFS\_S(1);

getch();

}