**Practical No. 08**

1. **Adjacency Matrix**

**Code:**

#include<iostream>

using namespace std;

#define max 100

class Graph{

public:

int a[max][max],i,n,k ;

char c[max],st[max],visited[max],ele;

int top;

Graph()

{

top = -1;

}

int isEmpty(char x)

{

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

{

if(st[i] == x)

{

return 1;

}

else

{

return 0;

}

}

}

void DFS()

{

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

{

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

if(a[i][j] == 1 && isEmpty(c[i]) == 0)

{

push(c[i]);

}

}

cout<<endl<<"A";

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

{

cout<<"\t"<<visited[i];

}

}

void getdata()

{

cout<<"Enter the number of vertices :-\n";

cin>>n;

cout<<"Enter each alphabets/digits :-\n";

c[n],visited[n];

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

{

cin>>c[i];

}

a[n][n];

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

{

visited[i] = '\0';

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

{

a[i][j] = 0;

}

}

}

void matrixdata()

{

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

{

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

{

if(i==j)

{

continue;

}

else

{

cout<<"Enter matrix data for "<<c[i]<<" x "<<c[j]<<" :-";

cin>>a[i][j];

}

}

}

}

void display()

{

cout<<"--------------------Adjacency Matrix--------------------"<<endl;

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

{

cout<<"\t"<<c[i];

}

cout<<endl;

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

{

cout<<c[i]<<"\t";

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

{

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

}

cout<<"\n";

}

}

void push(char x){

ele=x;

if(top>n-1)

{

cout<<"Overflow condition ";

}

else

{

top++;

k=0;

st[top] = visited[i] = ele;

cout<<"Element "<<ele<<" inserted at:"<<top;

k++;

}

cout<<endl;

}

int pop(){

if (top==-1)

{

cout<<"Underflow condition";

}

else

{

cout<<"The popped element is"<<st[top]<<endl;

top--;

}

}

};

int main()

{

int ch;

Graph g;

while(ch!=5)

{

cout<<"Enter the options \n1.Set Nodes \n2.Set Matrix Data \n3.Display \n4.DFS \n5.Exit "<<endl;

cout<<"Enter your choice";

cin>>ch;

switch(ch)

{

case 1: g.getdata();

break;

case 2: g.matrixdata();

break;

case 3: g.display();

break;

case 4: g.DFS();

break;

case 5: exit(0);

break;

default : cout<<"Invalid choice"<<endl;

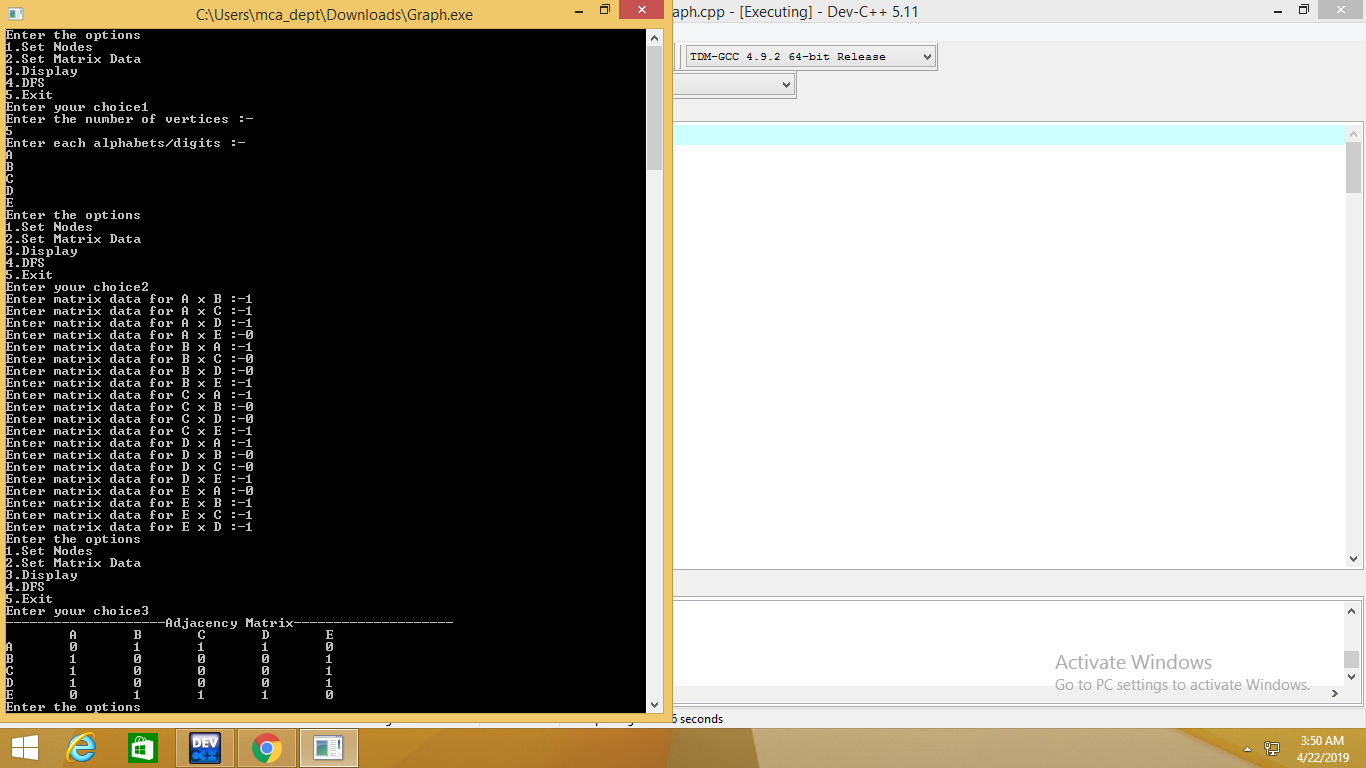
}

}

return 0;

}

**Screenshot:**

****

1. **Depth First Search**

**Code :**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

int mat[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10];

int main()

{

int m;

cout <<"enterno of vertices";

cin >> n;

cout <<"ente no of edges";

cin >> m;

cout <<"\nEDGES \n";

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

{

cin >>i>>j;

mat[i][j]=1;

}

cout <<"enter initial vertex";

cin >>v;

cout <<"ORDER OF VISITED VERTICES";

cout << v <<" ";

visited[v]=1;

k=1;

while(k<n)

{

for(j=n;j>=1;j--)

if(mat[v][j]!=0 && visited[j]!=1 && visit[j]!=1)

{

visit[j]=1;

stk[top]=j;

top++;

}

v=stk[--top];

cout<<v << " ";

k++;

visit[v]=0; visited[v]=1;

}

cout<<endl;

cout<<"--------------------Adjacency Matrix--------------------"<<endl;

cout<<endl;

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

{

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

{

cout<<mat[i][j]<<"\t";

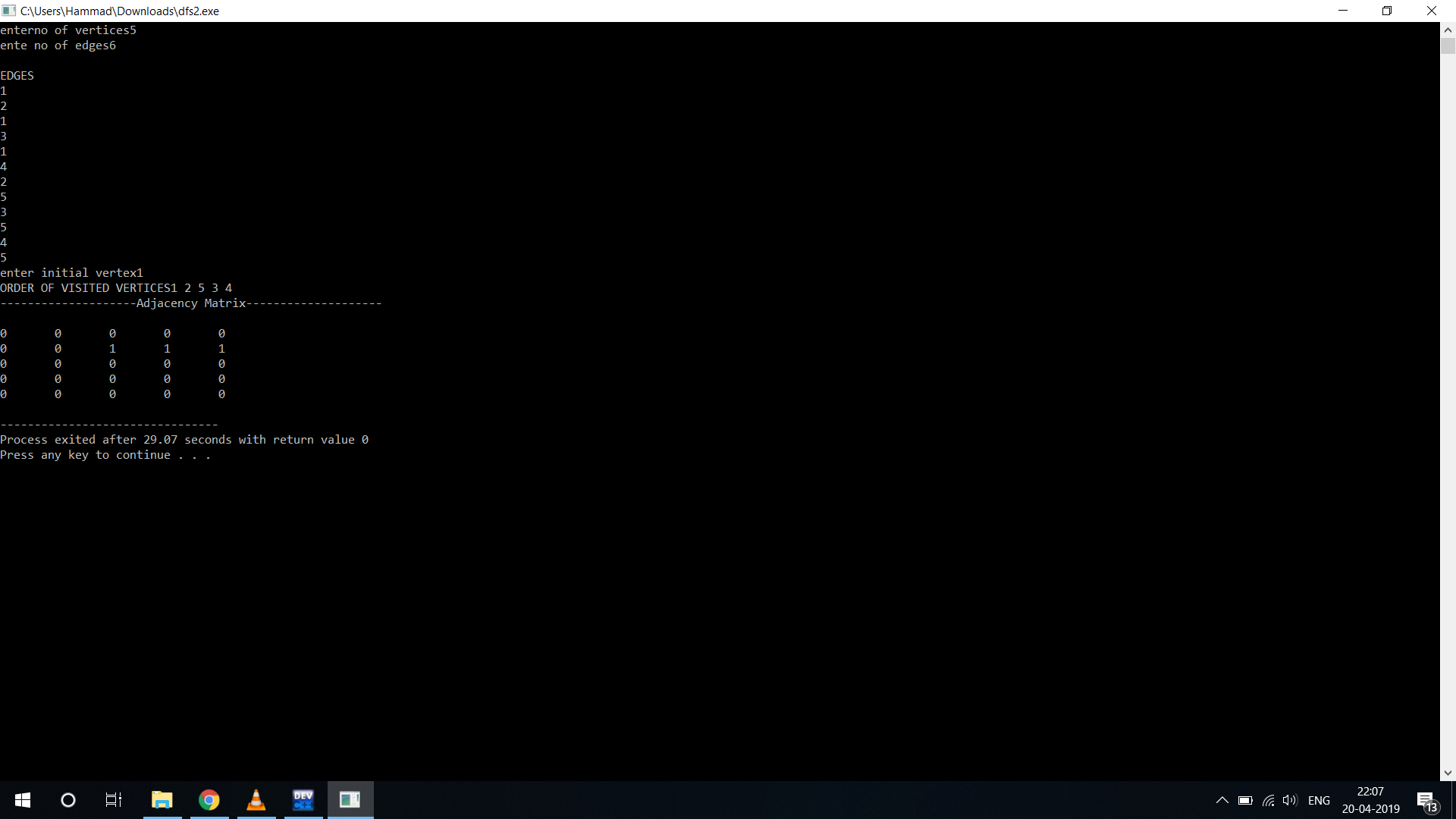
}

cout<<"\n";

}

}

**Screenshot :**



1. **Breadth First Search**

**Code :**

#include<iostream>

using namespace std;

class DFS{

public:

int noofnodes;

int matrix[100][100];

int queue[100], front, rear;

int mark[100];

void init(){

int n;

cout<<"Enter no of node : ";

cin>>n;

noofnodes=n;

front=-1;

rear=-1;

push(0);

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

for(int j=0;j<noofnodes;j++){

matrix[i][j]=0;

}

mark[i]=0;

}

}

void insert(){

int s, e;

cin>>s>>e;

matrix[--s][--e]=1;

matrix[e][s]=1;

}

void push(int s){

}

int isMarked(int a){

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

if(mark[i]==a)

return 1;

}

return 0;

}

void morePath()

{

int count=0;

queue[++front]=0;

mark[0]=1;

while(count<noofnodes)

{

int node = queue[++rear];

count++;

cout<<node+1<<endl;

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

{

if(matrix[node][i]==1)

if(mark[i]==0)

{

queue[++front]=i;

mark[i]=1;

}

}

}

}

void display(){

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

for(int j=0;j<noofnodes;j++){

cout<<matrix[i][j]<<"\t";

}

cout<<endl;

}

}

void menu(){

int task = 0;

while(task!=10){

cout<<"1. Init\n2. Insert\n3. Display\n4. Find a Path\n5. Exit\nEnter choice : ";

cin>>task;

switch(task){

case 1:

init();

break;

case 2:

insert();

break;

case 3:

display();

break;

case 4:

morePath();

break;

case 5:

task = 10;

break;

}

}

}

};

int main(){

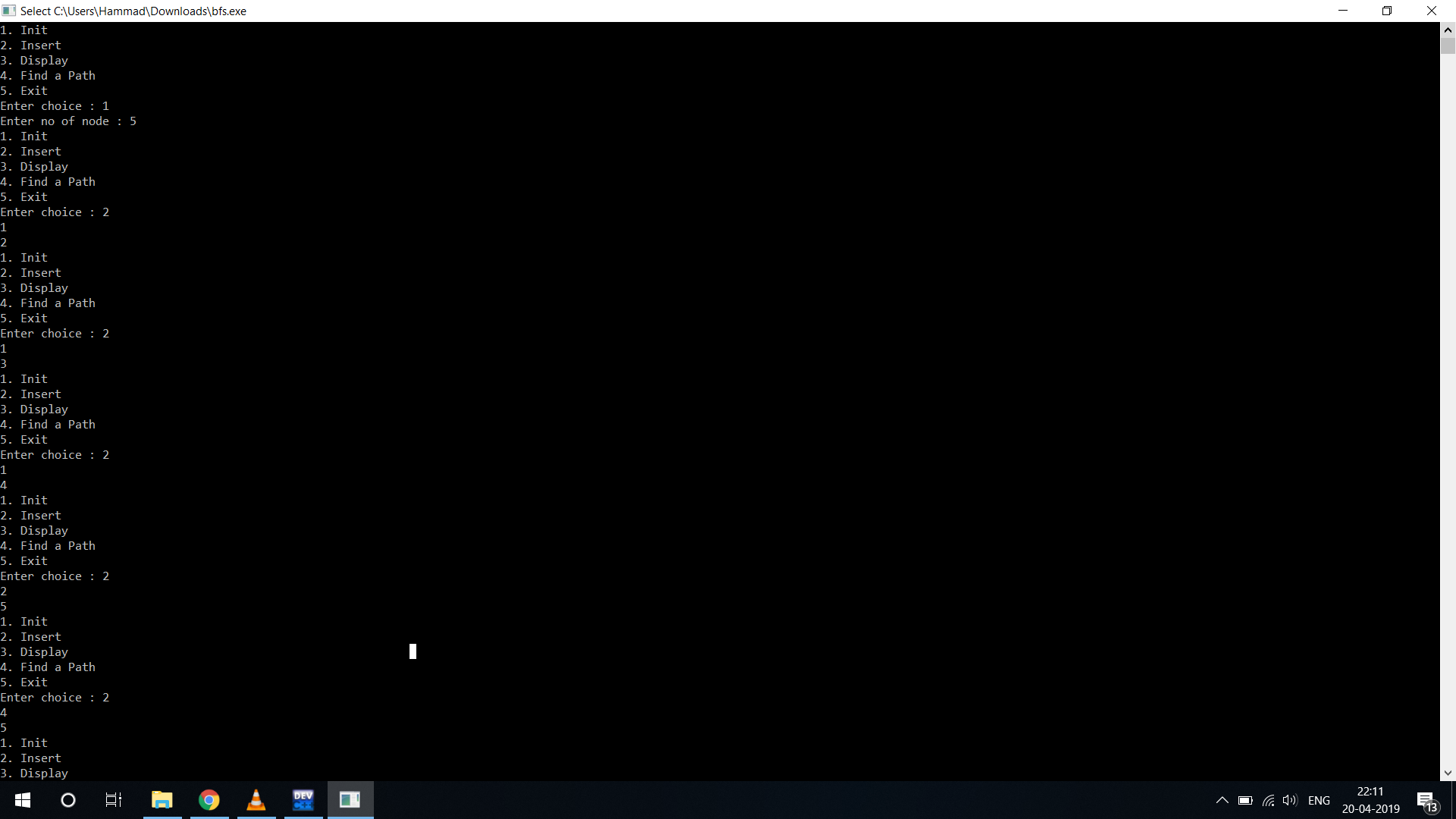
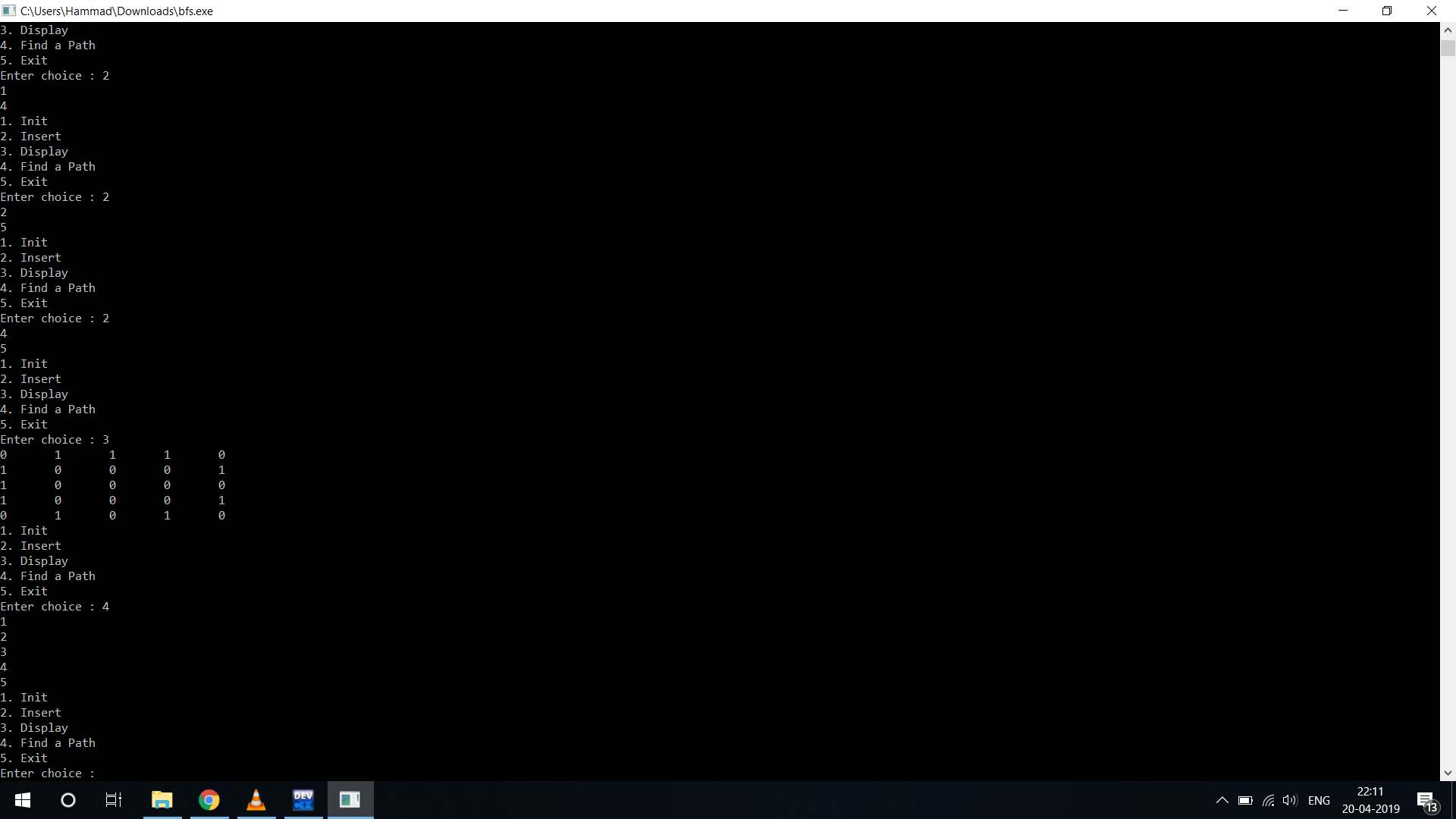
DFS a;

a.menu();

return 0;

}

**Screenshot :**

****