EXPERIMENT 8

- ASHISH KUMAR
- 2K18/SE/041

AIM:- Write a program to input graph matrix and perform DD path testing (triangle classification).

THEORY:- A graph matrix is a square matrix with one row and one column for every node of the graph.
The size of the matrix (number of rows and number of columns) is the number of nodes of the graph. Graph matrix is the tabular representation of a program graph. If we assign weight for every entry in the table, then this may be used for the identification of independent paths.
The simplest weight is 1, if there is a connection and 0 if there is no connection. A matrix with such weights is known as connection matrix.

DD path testing:-
A decision-to-decision path, or DD-path, is a path of execution (usually through a flow graph representing a program, such as a flow chart) between two decisions.

CODE:-

```
#include<bits/stdc++.h>

#include<iostream>

using namespace std;

void generate_path(int s,int n,vector<vector<int>> &v,vector<int> &visit,vector<int> path){

static int p;

 visit[s]=1;

 path.push_back(s);
```

```
bool e=true;
for(int i=s+1; i<n; i++)
{
if(v[s][i])
{
generate_path(i,n,v,visit,path);
 e=false;
}
}
if(e)
{
if(p<9)
cout<<++p<<")";
else
cout<<++p<<")";
for(unsigned int i=0; i<path.size(); i++)
{
cout<<" "<<path[i];
if(i!=path.size()-1)
cout<<"->";
}
if(p==1)
cout<<"\t\t"<<"40"<<"\t"<<"20"<<"\t"<<"50\tObtuse Angled Triangle";
else if(p==2)
```

```
cout<<"\t\t"<<"30"<<"\t"<<"40"<<"\t"<<"50\tRight Angled Triangle";
else if(p==3)
cout<<"\t\t"<<"20"<<"\t"<<"40"<<"\t"<<"50\tAcute Angled Triangle";
else if(p==9)
cout<<"\t\t\t"<<"30"<<"\t"<<"10"<<"\t"<<"10\tInvalid Triangle";
else if(p==15)
cout<<"\t\t\t\t\t"<<"30"<<"\t"<<"-1"<<"\t"<<"50\tInput Out of Range";
else
{
if(p==7 ||p==8)
cout<<"\t\t-\t-\t-\tInvalid Test Case";
else if(p==11)
cout<<"\t\t\t\t-\t-\t-\tInvalid Test Case";
else if(p==14)
cout<<"\t\t\t\t\t-\t-\t-\tInvalid Test Case";
else
cout<<"\t\t\t-\t-\t-\tInvalid Test Case";
}
cout<<endl;
}
visit[s]=0;
return;
}
```

```
int main()
{
 vector<vector<int>> graph(20,vector<int> (20,0));
 for(int i=0; i<=19; i++)
 {
 for(int j=0; j<=19; j++)
  graph[i][j]=0;
 }
 for(int i=0; i<=18; i++)
 {
 graph[i][i+1]=1;
 }
 graph[3][5]=1;
 graph[2][7]=1;
 graph[4][6]=1;
 graph[7][15]=1;
 graph[9][11]=1;
 graph[11][13]=1;
 graph[14][18]=1;
 graph[15][17]=1;
 graph[16][18]=1;
 graph[12][14]=1;
 graph[10][14]=1;
 graph[4][5]=0;
```

```
 graph[10][11]=0;
 graph[12][13]=0;
 graph[14][15]=0;
 graph[16][17]=0;
 cout<<"Input Graph Matrix:"<<endl;
 for(int i=0; i<=19; i++)
 {
 for(int j=0; j<=19; j++)
 cout<<graph[i][j]<<" ";
 cout<<endl;
 }
 cout<<endl;
 vector<int> path;
 vector<int> visit(20,0);
 cout<<"SNo"<<" "<<"  Path Used\t\t\t\t\t\t\tA\tB\tC\tExpected Output"<<endl;
 cout<<"-----------------------------------------------------------------------------------------------
----------\n";
 generate_path(0,20,graph,visit,path);
 return 0;
}
```

OUTPUT:-

```
C:\Users\Ashish\Desktop\dd_pathtesting.exe
Input Graph Matrix:
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SNo   Path Used                                                                A       B       C       Expected Output
--------------------------------------------------------------------------------------------------------------------------------
1) 0-> 1-> 2-> 3-> 4-> 6-> 7-> 8-> 9-> 10-> 14-> 18-> 19                      40      20      50      Obtuse Angled Triangle
2) 0-> 1-> 2-> 3-> 4-> 6-> 7-> 8-> 9-> 11-> 12-> 14-> 18-> 19                 30      40      50      Right Angled Triangle
3) 0-> 1-> 2-> 3-> 4-> 6-> 7-> 8-> 9-> 11-> 13-> 14-> 18-> 19                 20      40      50      Acute Angled Triangle
4) 0-> 1-> 2-> 3-> 4-> 6-> 7-> 15-> 16-> 18-> 19                              -       -       -       Invalid Test Case
5) 0-> 1-> 2-> 3-> 4-> 6-> 7-> 15-> 17-> 18-> 19                              -       -       -       Invalid Test Case
6) 0-> 1-> 2-> 3-> 5-> 6-> 7-> 8-> 9-> 10-> 14-> 18-> 19                              -       -       -       Invalid Test Case
7) 0-> 1-> 2-> 3-> 5-> 6-> 7-> 8-> 9-> 11-> 12-> 14-> 18-> 19                 -       -       -       Invalid Test Case
8) 0-> 1-> 2-> 3-> 5-> 6-> 7-> 8-> 9-> 11-> 13-> 14-> 18-> 19                 -       -       -       Invalid Test Case
9) 0-> 1-> 2-> 3-> 5-> 6-> 7-> 15-> 16-> 18-> 19                              30      10      10      Invalid Triangle
10) 0-> 1-> 2-> 3-> 5-> 6-> 7-> 15-> 17-> 18-> 19                             -       -       -       Invalid Test Case
11) 0-> 1-> 2-> 7-> 8-> 9-> 10-> 14-> 18-> 19                                 -       -       -       Invalid Test Case
12) 0-> 1-> 2-> 7-> 8-> 9-> 11-> 12-> 14-> 18-> 19                            -       -       -       Invalid Test Case
13) 0-> 1-> 2-> 7-> 8-> 9-> 11-> 13-> 14-> 18-> 19                            -       -       -       Invalid Test Case
14) 0-> 1-> 2-> 7-> 15-> 16-> 18-> 19                                         -       -       -       Invalid Test Case
15) 0-> 1-> 2-> 7-> 15-> 17-> 18-> 19                                         30      -1      50      Input Out of Range

--------------------------------
Process exited after 0.8089 seconds with return value 0
Press any key to continue . . .
```