**Name- ANUSHREE**

**Roll no- MCA139**

**University Registration no- TKM20MCA-2010**

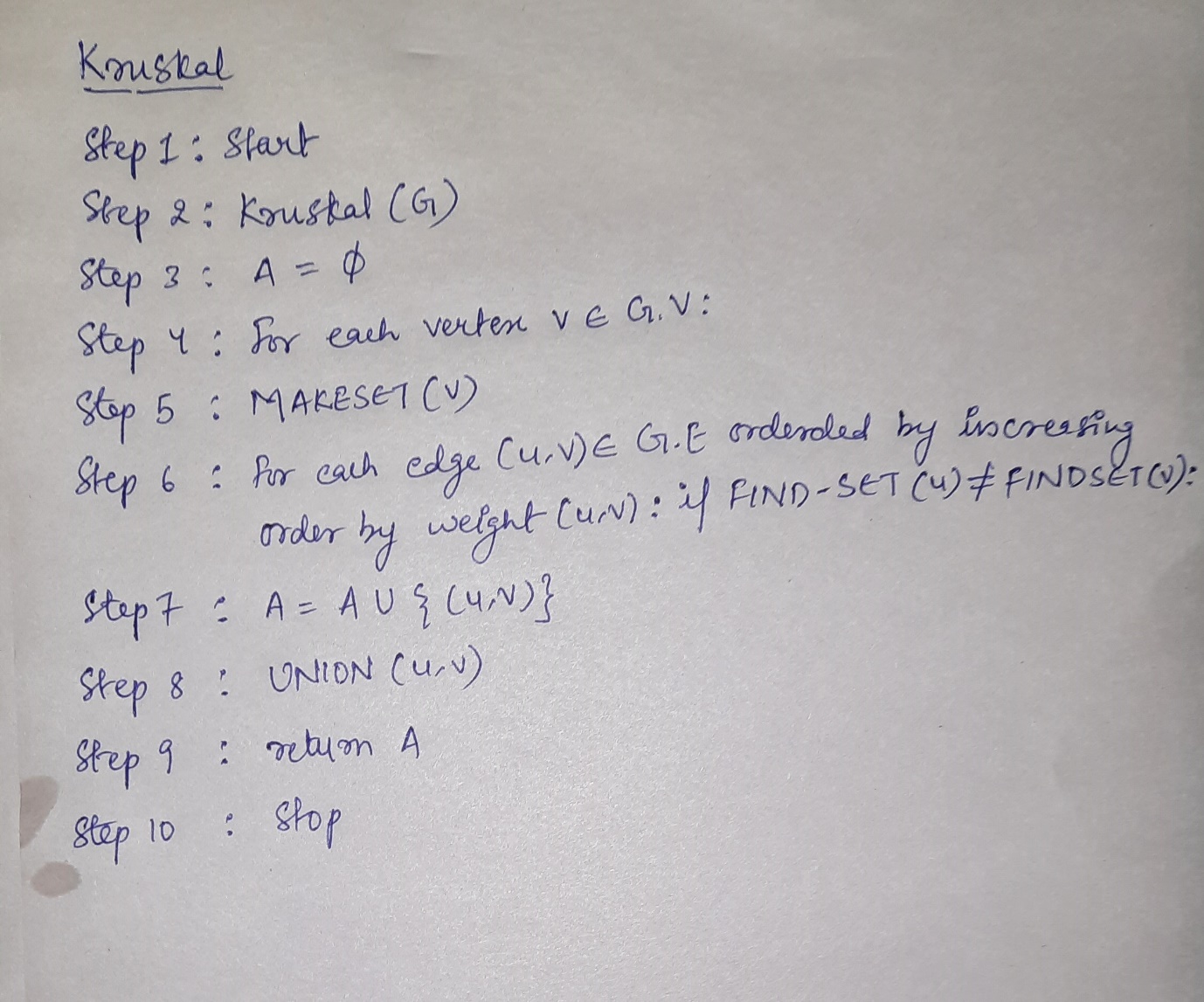
**EXTERNAL LAB EXAM (1ST SEMESTER)**

**DATA STRUCTURE LAB**

**QUESTION 1**

Develop a program to generate a minimum spanning tree using Kruskal algorithm for the given graph and compute the total cost.

**ALGORITHM**



**PROGRAM CODE**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int i,j,k,a,b,u,v,n,ne=1;

int min,mincost=0,cost[9][9],parent[9];

int find(int);

int uni(int,int);

void main()

{

printf("\n\tImplementation of Kruskal's algorithm\n");

printf("\nEnter the no. of vertices:");

scanf("%d",&n);

printf("\nEnter the cost adjacency matrix:\n");

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

{

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

{

scanf("%d",&cost[i][j]);

if(cost[i][j]==0)

cost[i][j]=999;

}

}

printf("The edges of Minimum Cost Spanning Tree are\n");

while(ne < n)

{

for(i=1,min=999;i<=n;i++)

{

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

{

if(cost[i][j] < min)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

}

}

u=find(u);

v=find(v);

if(uni(u,v))

{

printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);

mincost +=min;

}

cost[a][b]=cost[b][a]=999;

}

printf("\n\tMinimum cost = %d\n",mincost);

getch();

}

int find(int i)

{

while(parent[i])

i=parent[i];

return i;

}

int uni(int i,int j)

{

if(i!=j)

{

parent[j]=i;

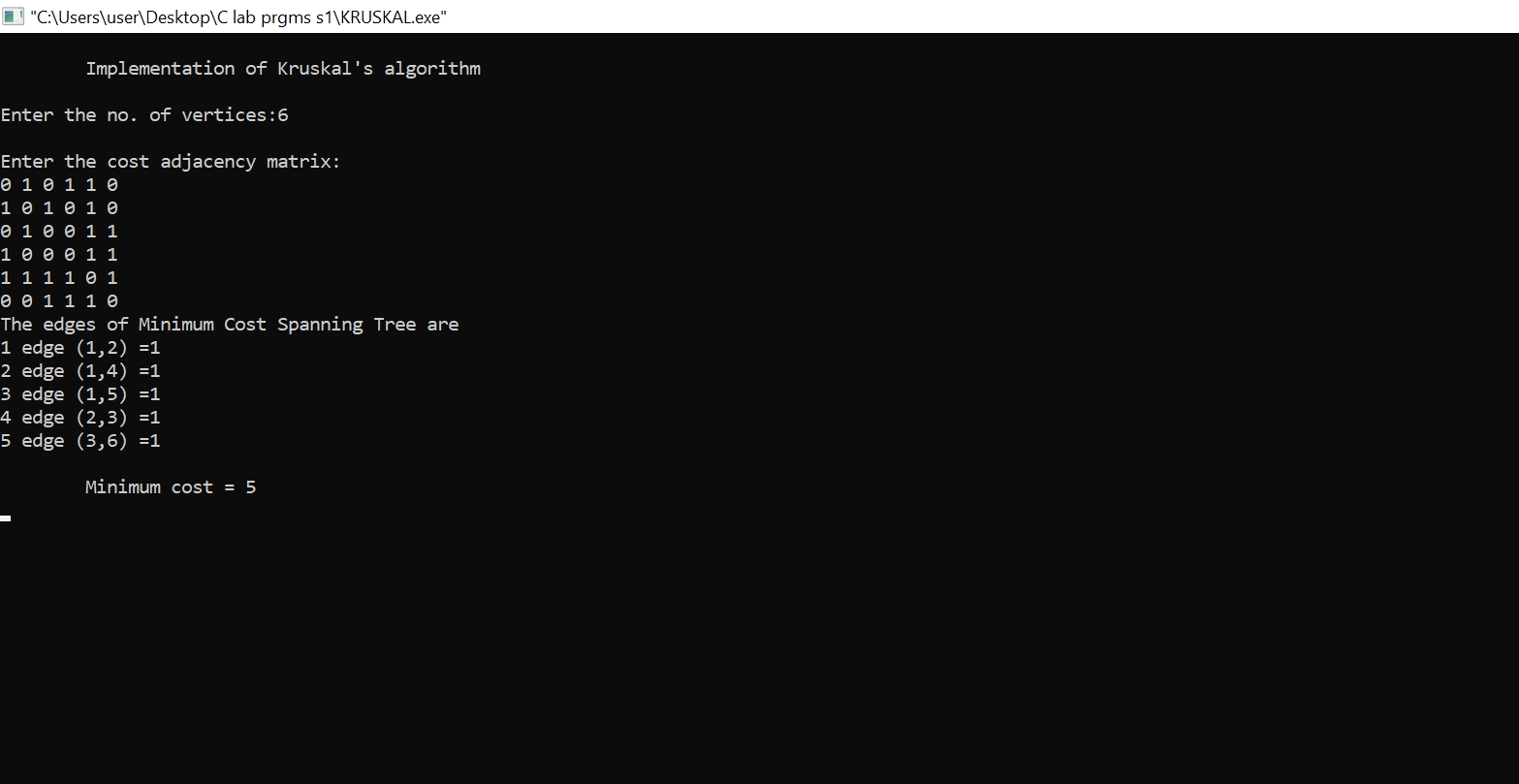
return 1;

}

return 0;

}

**OUTPUT**

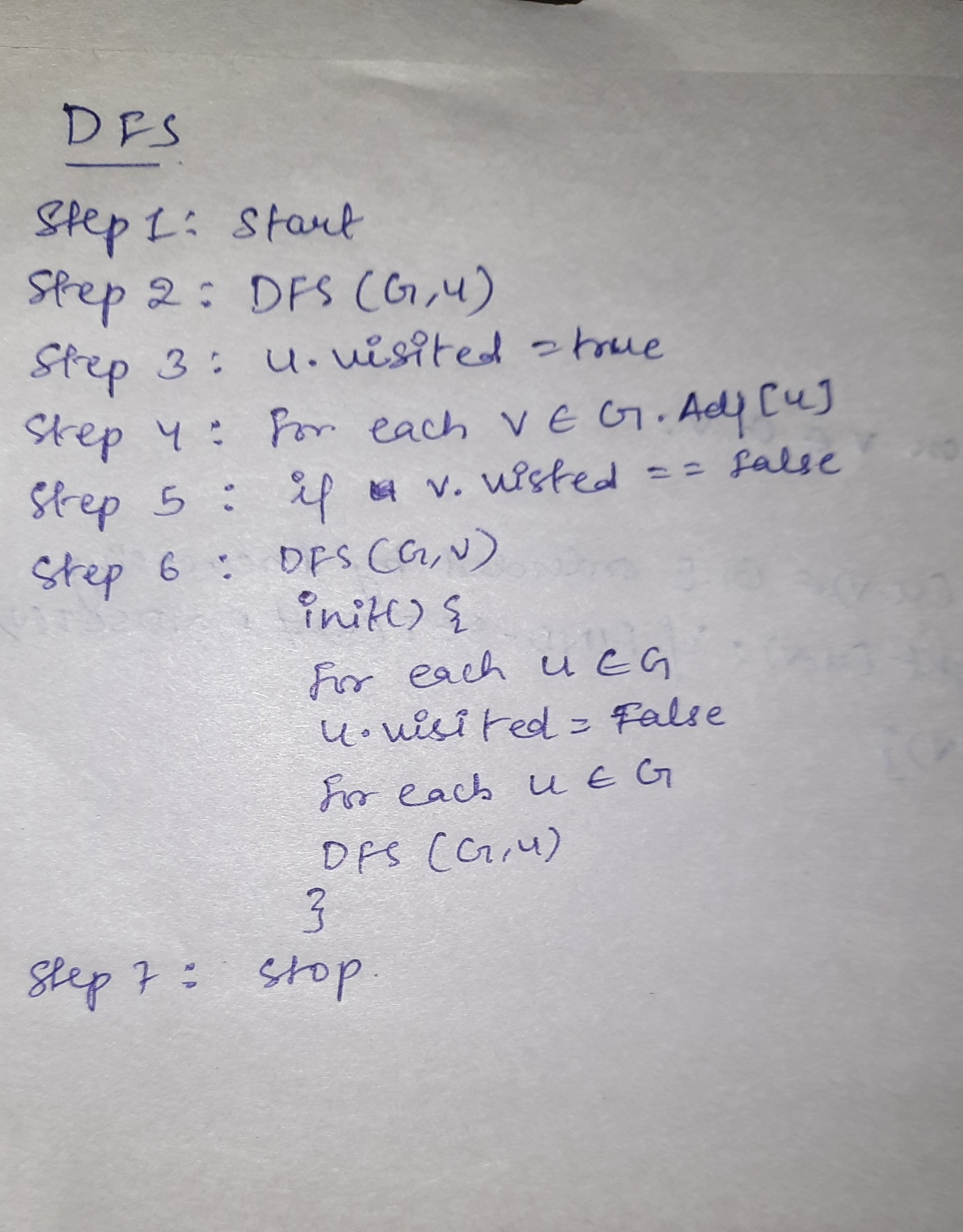


**QUESTION 2**

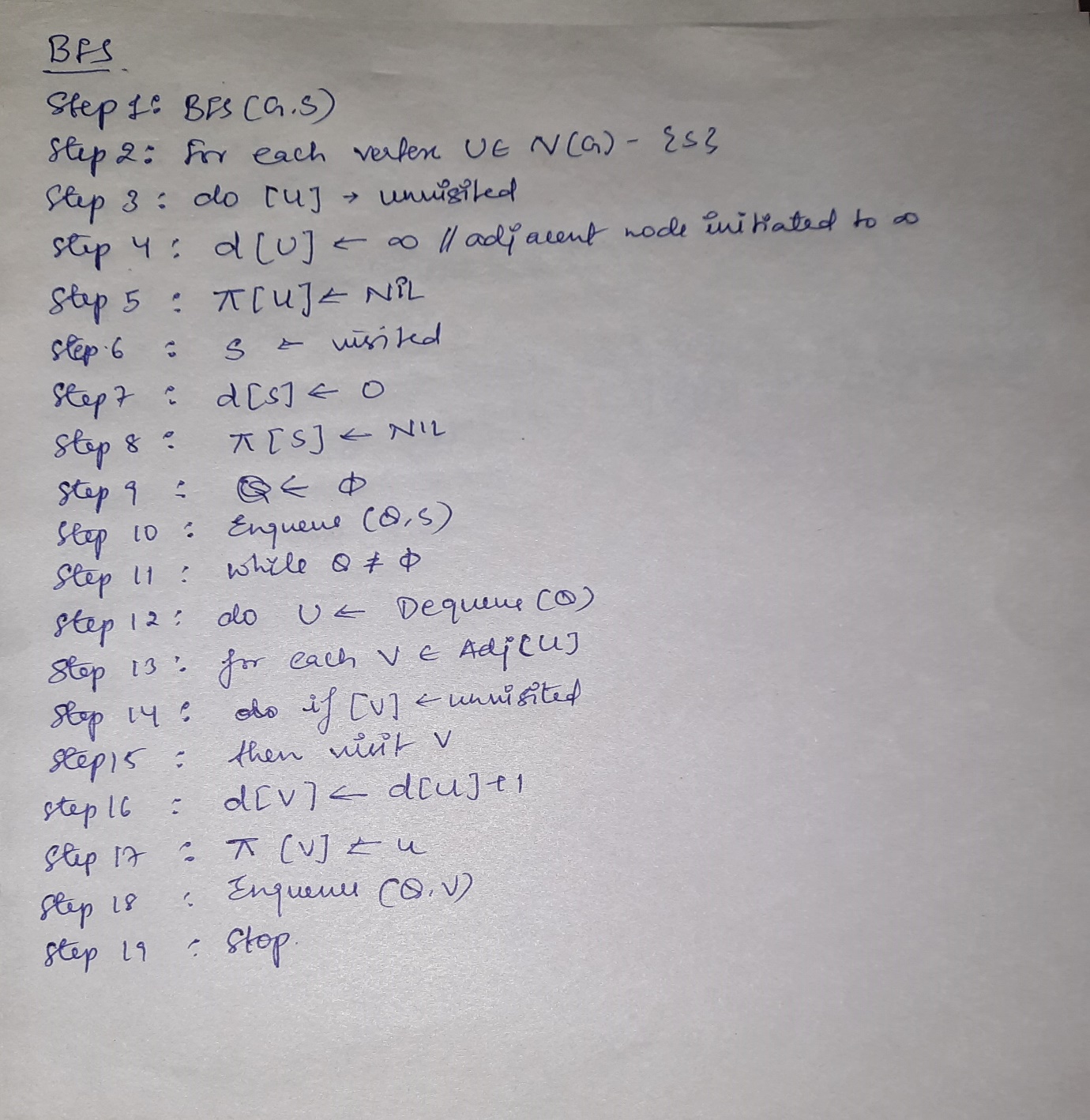
Develop a program to implement DFS and BFS.

**ALGORITHM**

**//DFS**



**//BFS**



**PROGRAM CODE**

**//DFS**

#include<stdio.h>

int a[20][20],reach[20],n; int dfs(int v)

{

int i; reach[v]=1;

for (i=1;i<=n;i++) if(a[v][i] && !reach[i])

{

printf("\n %d->%d",v,i); dfs(i);

}

}

int main()

{

int i,j,count=0;

printf("\*\*\*DFS Implementation\*\*\*");

printf("\n Enter number of vertices:"); scanf("%d",&n);

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

{

reach[i]=0;

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

a[i][j]=0;

}

printf("\n Enter the adjacency matrix:\n");

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

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

scanf("%d",&a[i][j]);

dfs(1);

printf("\n");

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

{

if(reach[i]) count++;

}

if(count==n)

printf("\n Graph is connected");

else

printf("\n Graph is not connected");

return 0;

}

**//BFS**

#include<stdio.h>

int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;

void bfs(int v)

{

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

if(a[v][i] && !visited[i])

q[++r] = i;

if(f <= r)

{

visited[q[f]] = 1;

bfs(q[f++]);

}

}

int main()

{

int v;

printf("\n Enter the number of vertices:");

scanf("%d", &n);

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

{

q[i] = 0;

visited[i] = 0;

}

printf("\n Enter graph data in matrix form:\n");

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

{

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

{

scanf("%d", &a[i][j]);

}

}

printf("\n Enter the starting vertex:");

scanf("%d", &v);

bfs(v);

printf("\n The node which are reachable are:\n");

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

{

if(visited[i])

printf("%d\t", i);

else

{

printf("\n Bfs is not possible. Not all nodes are reachable");

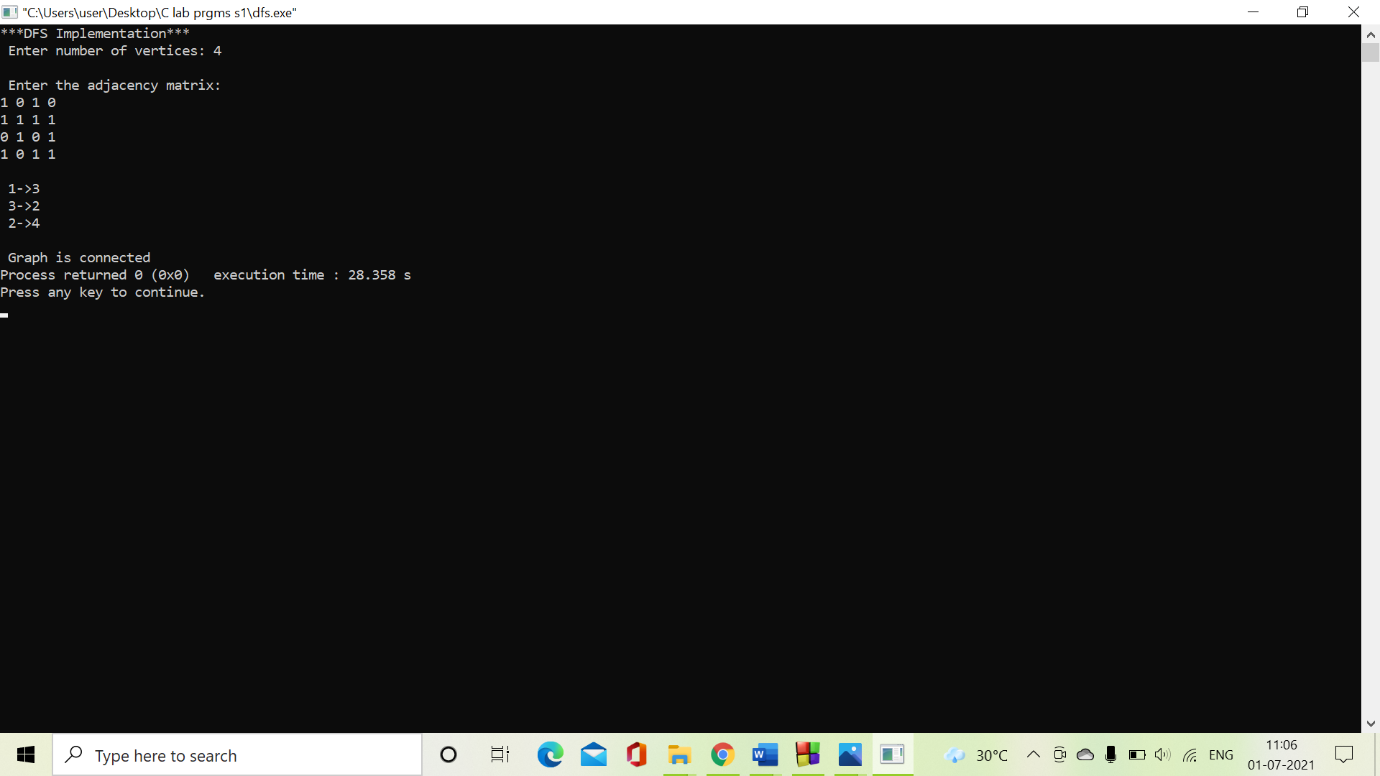
break;

}

}

}

**Output (DFS)**



**Output (BFS)**

