GANGA KRISHNAN.G

ROLL NO:18

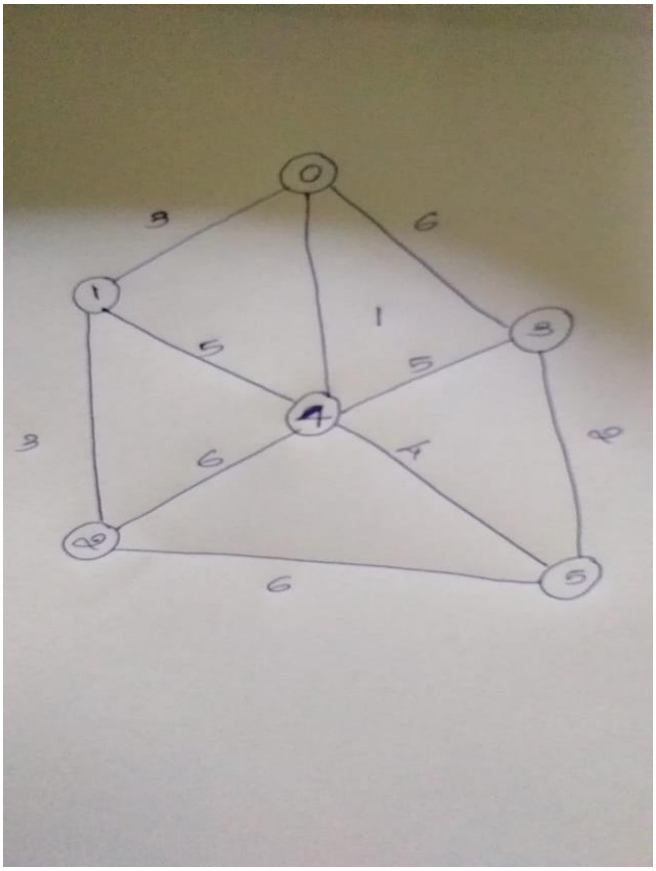
REGISTER NO : TKM20MCA-2018

S1 MCA

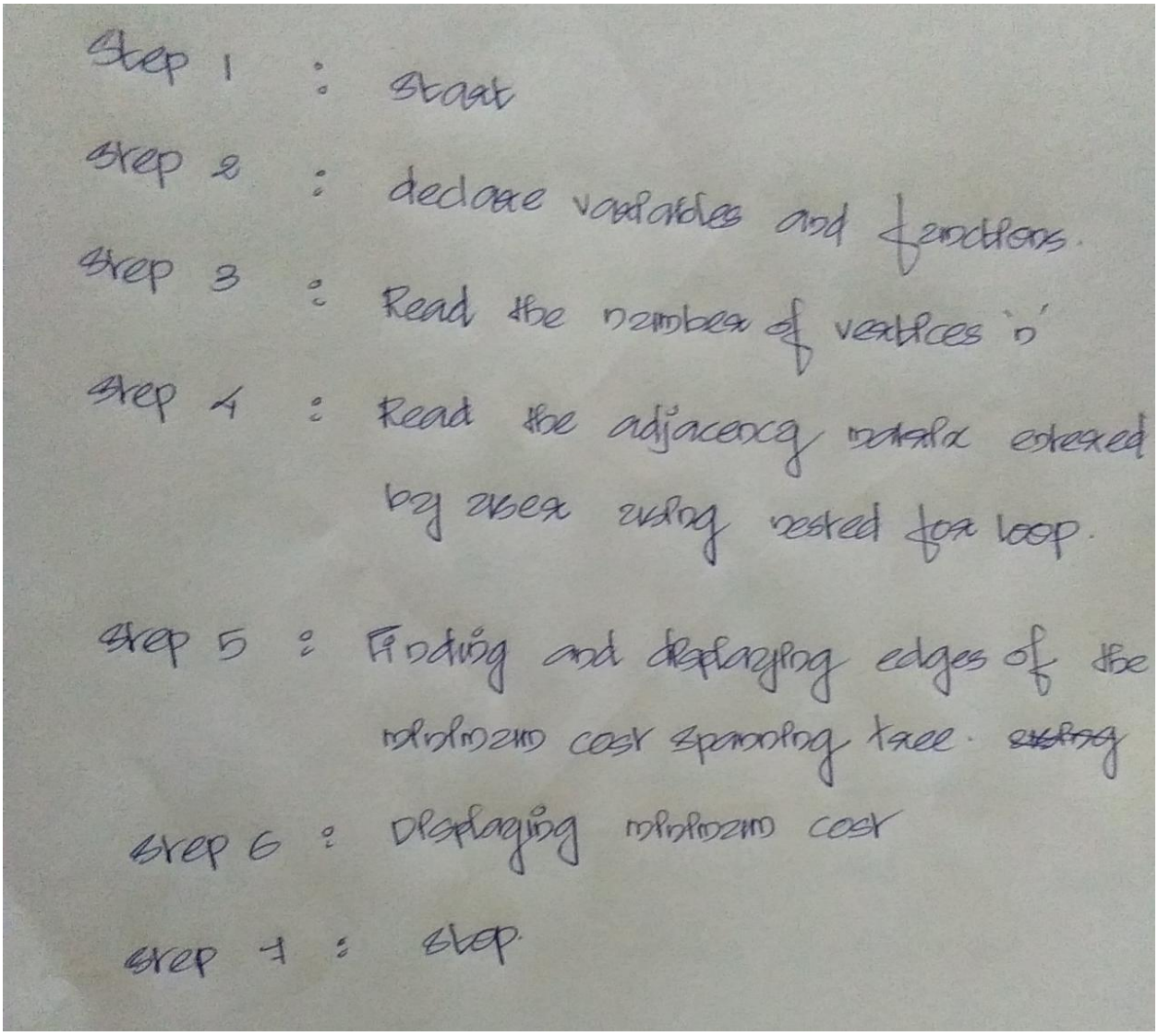
PROGRAM 1

AIM:

Develop a program to generate a minimum spanning tree using Kruskal's Algorithm for the given graph and compute total cost



ALGORITHM:



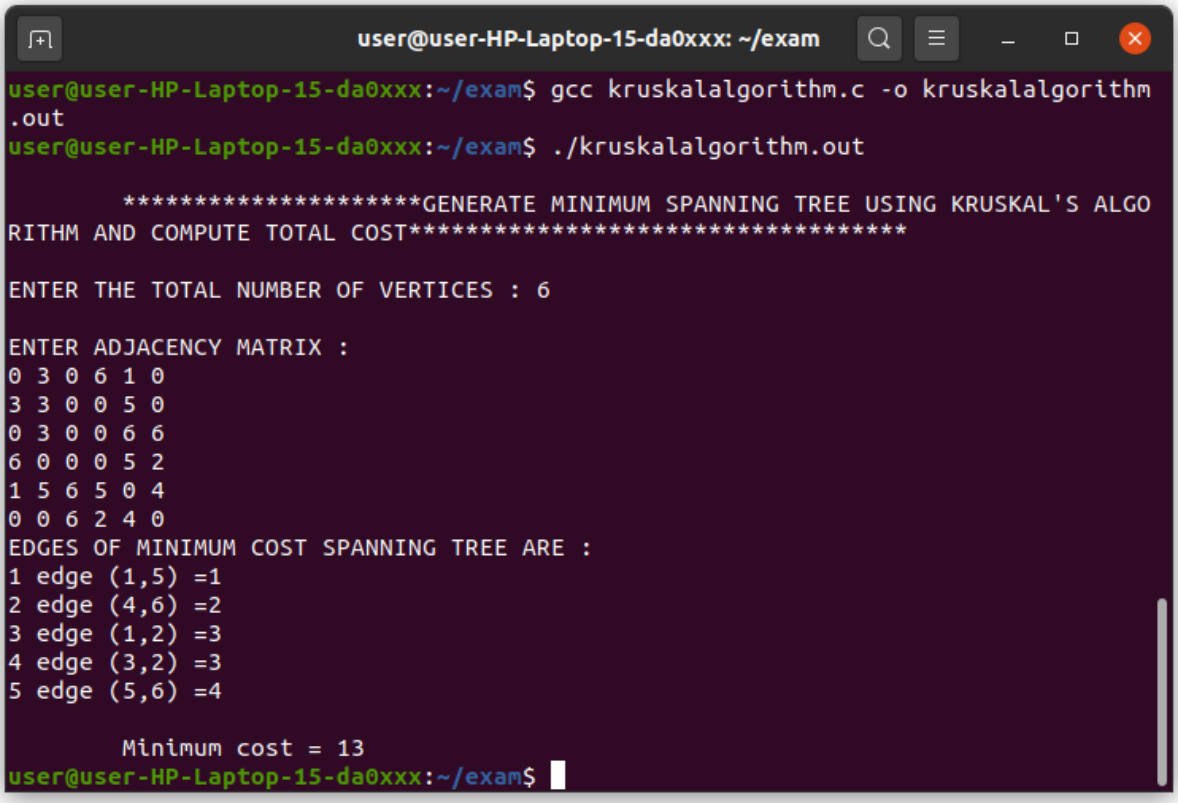
PROGRAM CODE:

```
#include<stdio.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\t**********GENERATE MINIMUM SPANNING TREE USING KRUSKAL'S ALGORITHM AND COMPUTE TOTAL COST**********\n");
	printf("\nENTER THE TOTAL NUMBER OF VERTICES : ");
	scanf("%d",&n);
	printf("\nENTER 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("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);

}
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;
}
```

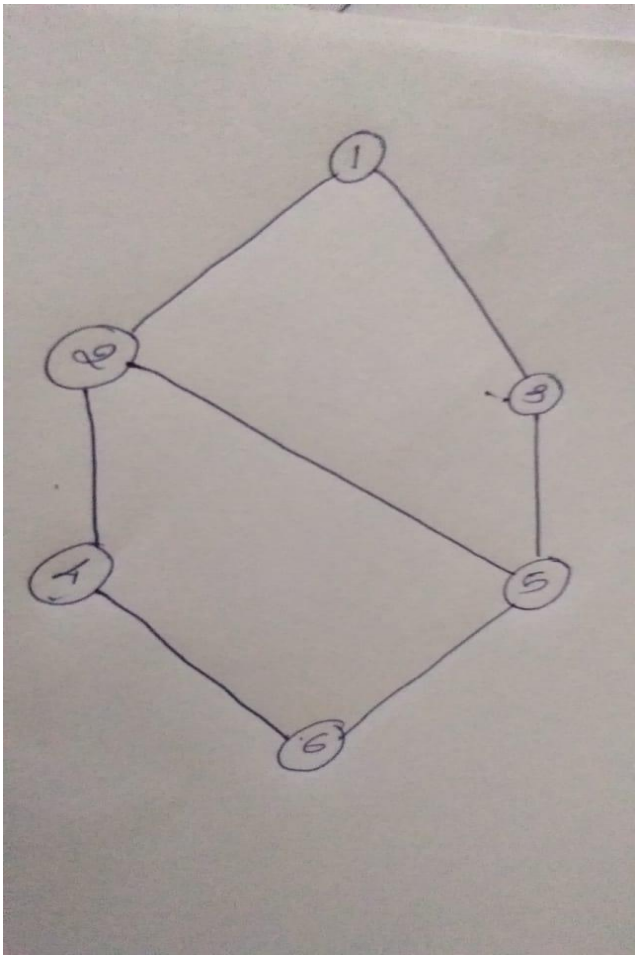
OUTPUT:


user@user-HP-Laptop-15-da0xxx: ~/exam
user@user-HP-Laptop-15-da0xxx:~/exam$ gcc kruskalalgorithm.c -o kruskalalgorithm
.out
user@user-HP-Laptop-15-da0xxx:~/exam$ ./kruskalalgorithm.out
********************GENERATE MINIMUM SPANNING TREE USING KRUSKAL'S ALGO
RITHM AND COMPUTE TOTAL COST**********
ENTER THE TOTAL NUMBER OF VERTICES : 6
ENTER ADJACENCY MATRIX :
0 3 0 6 1 0
3 3 0 0 5 0
0 3 0 0 6 6
6 0 0 0 5 2
1 5 6 5 0 4
0 0 6 2 4 0
EDGES OF MINIMUM COST SPANNING TREE ARE :
1 edge (1,5) =1
2 edge (4,6) =2
3 edge (1,2) =3
4 edge (3,2) =3
5 edge (5,6) =4
Minimum cost = 13
user@user-HP-Laptop-15-da0xxx:~/exam$

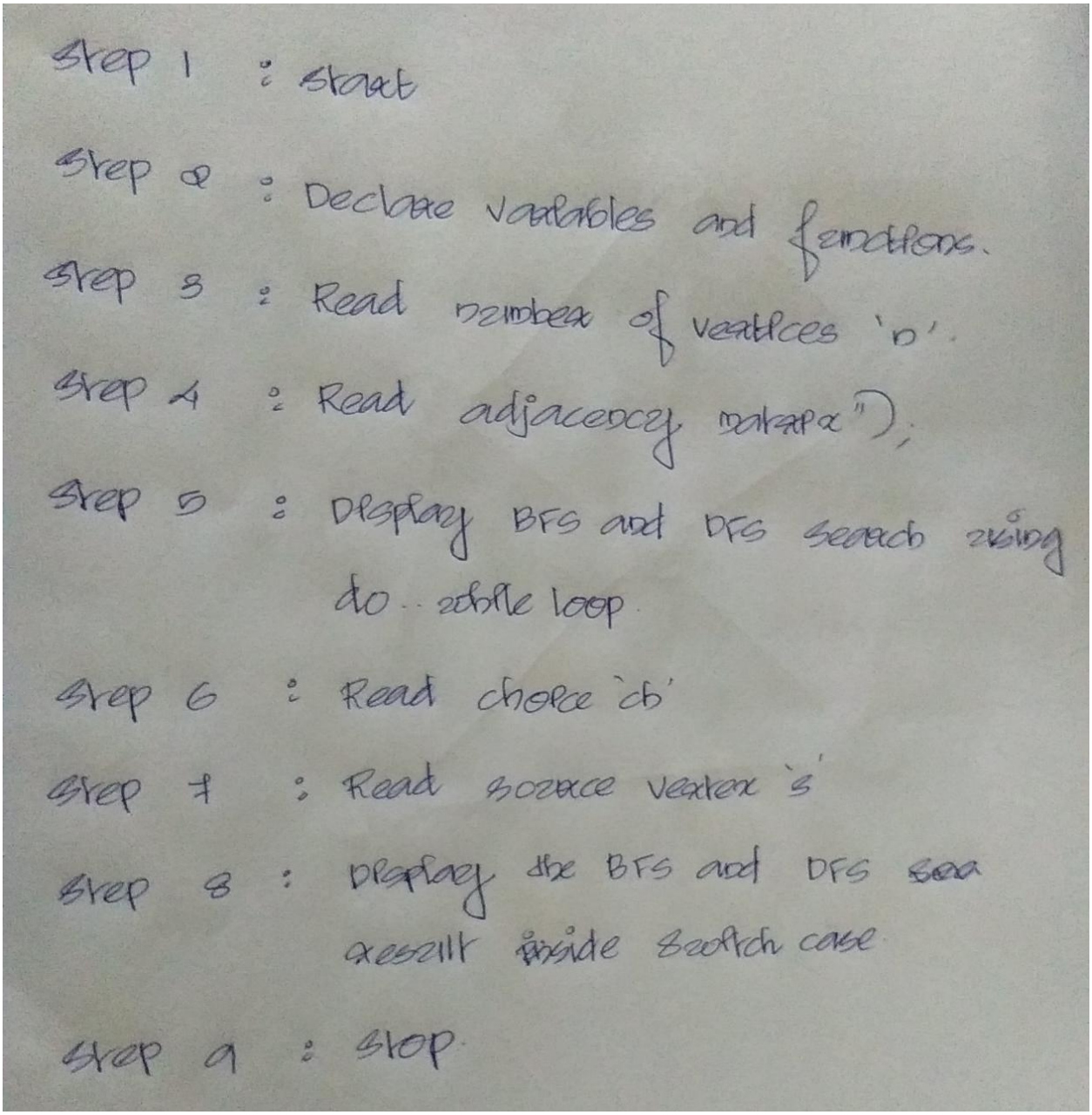
PROGRAM 2

AIM:

Develop a program to implement DFS and BFS



ALGORITHM:



PROGRAM CODE:

```
#include<stdio.h>


int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];

int delete();

void add(int item);
```

```
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();

void main()
{
int n,i,s,ch,j;
char c,dummy;
printf("ENTER THE NUMBER VERTICES ");
scanf("%d",&n);
printf("Enter the adjacency matrix:");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
scanf("%d",&a[i][j]);
}
printf("\n");
}

do
{
for(i=1;i<=n;i++)
vis[i]=0;
printf("\nMENU");
printf("\n1.B.F.S");
printf("\n2.D.F.S");
printf("\nENTER YOUR CHOICE");
```

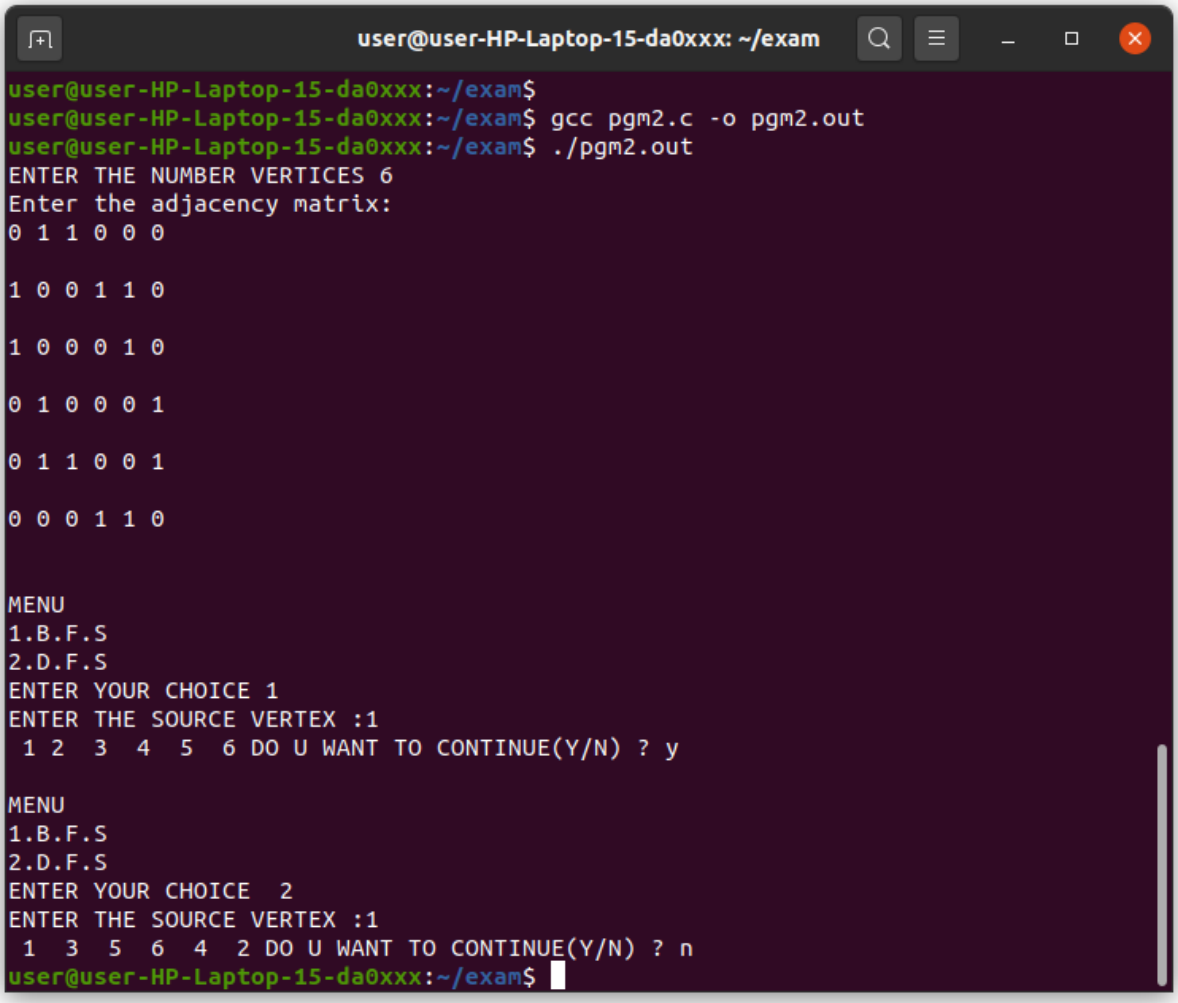
```
scanf("%d",&ch);
printf("ENTER THE SOURCE VERTEX :");
scanf("%d",&s);

switch(ch)
{
case 1:bfs(s,n);
break;
case 2:
dfs(s,n);
break;
}
printf("DO U WANT TO CONTINUE(Y/N) ? ");
scanf("%c",&dummy);
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
}
//**************BFS(breadth-first search) code**************//
void bfs(int s,int n)
{
int p,i;
add(s);
vis[s]=1;
p=delete();
if(p!=0)
printf(" %d",p);
while(p!=0)
{
for(i=1;i<=n;i++)
```

```
if((a[p][i]!=0)&&(vis[i]==0))
{
add(i);
vis[i]=1;
}
p=delete();
if(p!=0)
printf(" %d ",p);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
bfs(i,n);
}


void add(int item)
{
if(rear==19)
printf("QUEUE FULL");
else
{
if(rear==-1)
{
q[++rear]=item;
front++;
}
else
q[++rear]=item;
}
```

```
}
int delete()
{
int k;
if((front>rear)||(front==-1))
return(0);
else
{
k=q[front++];
return(k);
}
}


//***************DFS(depth-first search) code******************//
void dfs(int s,int n)
{
int i,k;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k);
while(k!=0)
{
for(i=1;i<=n;i++)
if((a[k][i]!=0)&&(vis[i]==0))
{
push(i);
```

```
vis[i]=1;
}
k=pop();
if(k!=0)
printf(" %d ",k);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);
}
void push(int item)
{
if(top==19)
printf("Stack overflow ");
else
stack[++top]=item;
}
int pop()
{
int k;
if(top==-1)
return(0);
else
{
k=stack[top--];
return(k);
}
}
```

OUTPUT:


user@user-HP-Laptop-15-da0xxx: ~/exam
user@user-HP-Laptop-15-da0xxx:~/exam$
user@user-HP-Laptop-15-da0xxx:~/exam$ gcc pgm2.c -o pgm2.out
user@user-HP-Laptop-15-da0xxx:~/exam$ ./pgm2.out
ENTER THE NUMBER VERTICES 6
Enter the adjacency matrix:
0 1 1 0 0 0
1 0 0 1 1 0
1 0 0 0 1 0
0 1 0 0 0 1
0 1 1 0 0 1
0 0 0 1 1 0
MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE 1
ENTER THE SOURCE VERTEX :1
1 2 3 4 5 6 DO U WANT TO CONTINUE(Y/N) ? y
MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE 2
ENTER THE SOURCE VERTEX :1
1 3 5 6 4 2 DO U WANT TO CONTINUE(Y/N) ? n
user@user-HP-Laptop-15-da0xxx:~/exam$

LINK TO GITHUB REPOSITORY :

https://github.com/ganga-17/datastructures.git