DFS BFS

Q) Write program to do the following:

a. Print all the nodes reachable from a given starting node in a digraph using BFS method.

b. Check whether a given graph is connected or not using DFS method.

```
#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;
printf("Enter The Number of Vertices ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("Enter 1 If %d Has A Node With %d Else 0 ",i,j);
scanf("%d",&a[i][j]);
}
}
```

```
printf("the adjacency matrix is\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf(" %d",a[i][j]);
}
printf("\n");
}
while(1){
{
for(i=1;i<=n;i++)
vis[i]=0;
printf("\nMENU\n1.BFS\n2.DFS\nenter choice");
scanf("%d",&ch);
printf("Enter source vertex:");
scanf("%d",&s);
switch(ch)
{
case 1:bfs(s,n);
break;
case 2:
dfs(s,n);
break;
}
}
}
```

```
}
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);
}
}
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);
}
}
```

```
Enter The Number of Vertices 4
Enter 1 If 1 Has A Node With 1 Else 0 0
Enter 1 If 1 Has A Node With 2 Else 0 1
Enter 1 If 1 Has A Node With 3 Else 0 1
Enter 1 If 1 Has A Node With 4 Else 0 1
Enter 1 If 2 Has A Node With 1 Else 0 0
Enter 1 If 2 Has A Node With 2 Else 0 0
Enter 1 If 2 Has A Node With 3 Else 0 0
Enter 1 If 2 Has A Node With 4 Else 0 1
Enter 1 If 3 Has A Node With 1 Else 0 0
Enter 1 If 3 Has A Node With 2 Else 0 0
Enter 1 If 3 Has A Node With 3 Else 0 0
Enter 1 If 3 Has A Node With 4 Else 0 0
Enter 1 If 4 Has A Node With 1 Else 0 0
Enter 1 If 4 Has A Node With 2 Else 0 0
Enter 1 If 4 Has A Node With 3 Else 0 1
Enter 1 If 4 Has A Node With 4 Else 0 0
the adjacency matrix is
 0 1 1 1
 0 0 0 1
 0 0 0 0
 0 0 1 0

MENU
1.BFS
2.DFS
enter choice1
Enter source vertex:1
 1 2  3  4
MENU
1.BFS
2.DFS
enter choice2
Enter source vertex:1
 1  4  3  2
MENU
1.BFS
2.DFS
enter choice
```

Write a c-program to implement DFS and BFS using adjecency matrix.

```
#include <stdio.h>
int q[20], top=-1, front=-1, rear=-1, a[20][20], vis[20]
f20, stack[20];
int delete();
void add (int item);
void bfs (int s, int n);
void dfs (int s, int n);
void push (int item);
void pop();
void main()
{
int n, i, s, ch, j;
char c, example;
Printf ("Enter the number of vertices");
scanf ("%d", &n);
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
Printf ("Enter 1 if %d has a node with %d else 0",i,j);
scanf ("%d", &a[i][j]);
}
}
printf ("The adjecency matrix is \n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
Printf ("%d",a[i][j]);
}
Printf ("\n");
}
```

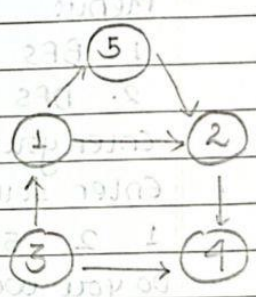
```
do
{
for (i=1;i<=n;i++)
vis[i]=0;
Printf("\n Menue");
Printf("\n1. BFS");
Printf("\n2. DFS");
Printf("\n Enter your choice");
scanf("%d", &ch);
Printf("Enter the source vertex:");
scanf("%d", &s);
switch(ch)
{
case1: BFS(s,n);
break;
case2 : DFS(s,n);
break;
}
Printf("Do you want to continue (Y/N)?");
scanf("%c", &example);
scanf("%c", &c);
}
while((c=='y')||(c=='Y'));
}
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);
}
}
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.




```
Enter no. of vertices: 5
Enter 1 if 1 has a node with 1 else 0 : 0
Enter 1 if 1 has a node with 2 else 0 : 1
   "    "    "    3 else 0 : 0
   "    "    "    4 else 0 : 0
   "    "    "    5 else 0 : 1
Enter 1 if 2 has a node with 1 else 0 : 0
Enter 1 if 2 has a node with 2 else 0 : 0
   "    "    "    3 else 0 : 0
   "    "    "    4 else 0 : 1
   "    "    "    5 else 0 : 0
Enter 1 if 3 has a node with 1 else 0 : 1
Enter 1 if 3 has a node with 2 else 0 : 0
   "    "    "    3 else 0 : 0
   "    "    "    4 else 0 : 1
   "    "    "    5 else 0 : 0
Enter 1 if 4 has a node with 1 else 0 : 0
Enter 1 if 4 has a node with 2 else 0 : 0
   "    "    "    3 else 0 : 0
```

" " " 4 else 0: 0
" " " 5 else 0: 0

Enter 1 if 5 Has a node with 1 else 0: 0
Enter 1 if 5 Has a node with 2 else 0: 1
" " " 3 else 0: 0
" " " 4 else 0: 0
" " " 5 else 0: 0

The adjacency matrix is

0 1 0 0 1
0 0 0 1 0
1 0 0 1 0
0 0 0 0 0
0 1 0 0 0

Menue
1. BFS
2. DFS
Enter your choice 1
Enter source vertex: 1
1 2 5 4 3
Do you want to continue (Y/N)? Y
Menue
1. BFS
2. DFS
Enter your choice 2
Enter source vertex 1
1 5 2 4 3