Graph Traversals

```c
#ifndef Queue_h
#define Queue_h
#include <stdlib.h>

struct Node
{
    int data;
    struct Node *next;

}*front=NULL,*rear=NULL;

void enqueue(int x)
{
    struct Node *t;
    t=(struct Node*)malloc(sizeof(struct Node));
    if(t==NULL)
        printf("Queue is FUll\n");
    else
    {
        t->data=x;
        t->next=NULL;
        if(front==NULL)
            front=rear=t;
        else
        {
            rear->next=t;
            rear=t;
        }
    }

}

int dequeue()
{
    int x=-1;
    struct Node* t;

    if(front==NULL)
        printf("Queue is Empty\n");
    else
    {
        x=front->data;
        t=front;
```

```
        front=front->next;
        free(t);
    }
    return x;
}

int isEmpty()
{
    return front==NULL;
}

#endif /* Queue_h */


#include <stdio.h>
#include "Queue.h"

void BFS(int G[][7],int start,int n)
{
    int i=start,j;

    int visited[7]={0};

    printf("%d ",i);
    visited[i]=1;
    enqueue(i);

    while(!isEmpty())
    {
        i=dequeue();
        for(j=1;j<n;j++)
        {
            if(G[i][j]==1 && visited[j]==0)
            {
                printf("%d ",j);
                visited[j]=1;
                enqueue(j);
            }
        }
    }


}



void DFS(int G[][7],int start,int n)
{
```

```
    static int visited[7]={0};
    int j;

    if(visited[start]==0)
    {
        printf("%d ",start);
        visited[start]=1;

        for(j=1;j<n;j++)
        {
            if(G[start][j]==1 && visited[j]==0)
                DFS(G,j,n);
        }
    }
}


int main()
{
    int G[7][7]={{0,0,0,0,0,0,0},
                 {0,0,1,1,0,0,0},
                 {0,1,0,0,1,0,0},
                 {0,1,0,0,1,0,0},
                 {0,0,1,1,0,1,1},
                 {0,0,0,0,1,0,0},
                 {0,0,0,0,1,0,0}};

    DFS(G,4,7);

    return 0;
}
```