EX NO 11 IMPLEMENTATION OF DFS AND BFS

Implementation of BFS

```
#include <stdio.h>
#include <stdlib.h>

#define MAX_SIZE 7

int queue[MAX_SIZE];
int front = -1, rear = -1;

int isEmpty() { return front == -1 && rear == -1; }

int isFull() { return rear == MAX_SIZE - 1; }

void enqueue(int val) {
    if (!isFull()) {
        if (isEmpty()) {
            front = rear = 0;
        } else {
            rear = (rear + 1) % MAX_SIZE;
        }
        queue[rear] = val;
    } else {
        printf("\nQUEUE IS FULL!\n");
    }
}

int dequeue() {
    if (!isEmpty()) {
        int val = queue[front];
        if (front == rear) {
            front = rear = -1;
        } else {
            front = (front + 1) % MAX_SIZE;
        }
        return val;
    } else {
        printf("\nQUEUE IS EMPTY!\n");
        return -1;
    }
```

```
}

int visited[MAX_SIZE] = {0};

int main() {
    int g[MAX_SIZE][MAX_SIZE] = {
        {0, 1, 1, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 0},
        {0, 0, 0, 1, 0, 1, 0},
        {1, 1, 0, 0, 0, 0, 1},
        {0, 1, 0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0, 0, 1},
        {0, 0, 0, 0, 1, 0, 0}
    };

    int i = 0;

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

    enqueue(i);

    while (!isEmpty()) {
        int i = dequeue();

        for (int j = 0; j < MAX_SIZE; j++) {
            if (g[i][j] && !visited[j]) {
                visited[j] = 1;
                printf("%d -> ", j);
                enqueue(j);
            }
        }
    }

    return 0;
}
```

Implementation of DFS

```
#include<stdio.h>
#include<stdlib.h>
#define size 7
int s[size];
```

```
int top=-1;
int pop();
void push(int);

void main(){
   int
g[size][size]={{0,1,1,0,0,0,0},{0,0,0,0,0,0,0},{0,0,0,1,0,1,0},{1,1,0,0,0,0,1},{0,1,0,0,0,0,0},{0,0,0,0,
0,0,1},{0,0,0,0,1,0,0}};
   int visited[size]={0};
   int j,i=0;

   while(i>-1 && i<size)
   {
      if(visited[i]!=1)
      {
      printf("%d ",i);
      visited[i]=1;
      }
      for(i,j=0;j<size;j++)
      {
         if(g[i][j]==1 && visited[j]!=1){
         push(j);
         }
      }
      i=pop();
   }
}



void push(int data)
{
      top=top+1;
      s[top]=data;

}

int pop()
{
      int temp;
      temp=s[top];
      top=top-1;
      return temp;
```

}

OUTPUT:

```
0 1 3 4 5 6 2
```