Modular BFS and DFS in C (No Comments)

File 1: ds.h

```c
#ifndef DS_H
#define DS_H

#include <stdio.h>
#define MAX 10

typedef struct {
    int items[MAX];
    int front, rear;
} Queue;

void initQueue(Queue *q) {
    q->front = q->rear = -1;
}

int isQueueEmpty(Queue *q) {
    return q->front == -1;
}

void enqueue(Queue *q, int value) {
    if (q->rear == MAX - 1) {
        printf("Queue Overflow!\n");
        return;
    }
    if (q->front == -1)
        q->front = 0;
    q->items[++(q->rear)] = value;
}

int dequeue(Queue *q) {
    if (isQueueEmpty(q)) {
        printf("Queue Underflow!\n");
        return -1;
    }
    int value = q->items[q->front];
    if (q->front == q->rear)
        q->front = q->rear = -1;
    else
        q->front++;
    return value;
}

typedef struct {
    int items[MAX];
    int top;
} Stack;

void initStack(Stack *s) {
    s->top = -1;
}

int isStackEmpty(Stack *s) {
    return s->top == -1;
}

void push(Stack *s, int value) {
    if (s->top == MAX - 1) {
        printf("Stack Overflow!\n");
        return;
    }
    s->items[++(s->top)] = value;
}

int pop(Stack *s) {
    if (isStackEmpty(s)) {
        printf("Stack Underflow!\n");
        return -1;
    }
    return s->items[(s->top)--];
}

#endif
```

File 2: main.c

```
#include <stdio.h>
#include "ds.h"

void BFS(int graph[MAX][MAX], int vertices, int startVertex) {
    int visited[MAX] = {0};
    Queue q;
    initQueue(&q);
    visited[startVertex] = 1;
    enqueue(&q, startVertex);
    printf("\nBFS Traversal: ");
    while (!isQueueEmpty(&q)) {
        int currentVertex = dequeue(&q);
        printf("%d ", currentVertex);
        for (int i = 0; i < vertices; i++) {
            if (graph[currentVertex][i] && !visited[i]) {
                visited[i] = 1;
                enqueue(&q, i);
            }
        }
    }
    printf("\n");
}

void DFS(int graph[MAX][MAX], int vertices, int startVertex) {
    int visited[MAX] = {0};
    Stack s;
    initStack(&s);
    push(&s, startVertex);
    printf("\nDFS Traversal: ");
    while (!isStackEmpty(&s)) {
        int currentVertex = pop(&s);
        if (!visited[currentVertex]) {
            printf("%d ", currentVertex);
            visited[currentVertex] = 1;
        }
        for (int i = vertices - 1; i >= 0; i--) {
            if (graph[currentVertex][i] && !visited[i]) {
                push(&s, i);
            }
        }
    }
    printf("\n");
}

int main() {
    int vertices = 5;
    int graph[MAX][MAX] = {
        {0, 1, 1, 0, 0},
        {1, 0, 1, 1, 0},
        {1, 1, 0, 0, 1},
        {0, 1, 0, 0, 1},
        {0, 0, 1, 1, 0}
    };
    int startVertex = 0;
    printf("Graph (Adjacency Matrix):\n");
    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            printf("%d ", graph[i][j]);
        }
        printf("\n");
    }
    BFS(graph, vertices, startVertex);
    DFS(graph, vertices, startVertex);
    return 0;
}
```