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

#define MAX_CITIES 100

// Structure to represent a graph
struct Graph {
    int vertices;
    int adjacencyMatrix[MAX_CITIES][MAX_CITIES];
};

// Function to initialize the graph
void initializeGraph(struct Graph *G, int vertices) {
    G->vertices = vertices;
    for (int i = 0; i < MAX_CITIES; i++) {
        for (int j = 0; j < MAX_CITIES; j++) {
            G->adjacencyMatrix[i][j] = 0;
        }
    }
}

// Function to add an edge to the graph
void addEdge(struct Graph *G, int source, int destination) {
    if (source >= 0 && source < G->vertices && destination >= 0
&& destination < G->vertices) {
        G->adjacencyMatrix[source][destination] = 1;
    } else {
        printf("Invalid edge: %d -> %d\n", source,
destination);
    }
}

// Depth-First Search (DFS) function
void DFS(struct Graph *G, int start, int visited[]) {
    printf("%d ", start);
    visited[start] = 1;

    for (int i = 0; i < G->vertices; i++) {
        if (G->adjacencyMatrix[start][i] == 1 && !visited[i]) {
            DFS(G, i, visited);
        }
    }
}

// Breadth-First Search (BFS) function
```

```
void BFS(struct Graph *G, int start) {
    int queue[MAX_CITIES];
    int front = 0, rear = 0;
    int visited[MAX_CITIES] = {0};

    printf("%d ", start);
    visited[start] = 1;
    queue[rear++] = start;

    while (front < rear) {
        int current = queue[front++];
        for (int i = 0; i < G->vertices; i++) {
            if (G->adjacencyMatrix[current][i] == 1 && !
visited[i]) {
                printf("%d ", i);
                visited[i] = 1;
                queue[rear++] = i;
            }
        }
    }
}

int main() {
    struct Graph G;
    int n, start;

    // Input the number of cities
    printf("Enter the number of cities: ");
    scanf("%d", &n);

    // Initialize the graph
    initializeGraph(&G, n);

    // Input the edges (connections between cities)
    int edges;
    printf("Enter the number of edges: ");
    scanf("%d", &edges);

    for (int i = 0; i < edges; i++) {
        int source, destination;
        printf("Enter edge #%d (source destination): ", i + 1);
        scanf("%d %d", &source, &destination);
        addEdge(&G, source, destination);
    }
```

```
    // Input the starting node for DFS/BFS
    printf("Enter the starting node for DFS/BFS: ");
    scanf("%d", &start);

    printf("DFS traversal starting from node %d: ", start);
    int visitedDFS[MAX_CITIES] = {0};
    DFS(&G, start, visitedDFS);
    printf("\n");

    printf("BFS traversal starting from node %d: ", start);
    BFS(&G, start);
    printf("\n");

    return 0;
}
```