Write a C program on BFS, DFS

#include <stdio.h>

#include <stdbool.h>

#define MAX\_VERTICES 100

bool visited[MAX\_VERTICES];

typedef struct {

int adjacencyMatrix[MAX\_VERTICES][MAX\_VERTICES];

int numVertices;

} Graph;

void initializeGraph(Graph\* graph, int numVertices) {

graph->numVertices = numVertices;

for (int i = 0; i < numVertices; i++) {

for (int j = 0; j < numVertices; j++) {

graph->adjacencyMatrix[i][j] = 0;

}

}

}

void addEdge(Graph\* graph, int src, int dest) {

if (src >= 0 && src < graph->numVertices && dest >= 0 && dest < graph->numVertices) {

graph->adjacencyMatrix[src][dest] = 1;

graph->adjacencyMatrix[dest][src] = 1; // Uncomment for undirected graph

} else {

printf("Invalid edge.\n");

}

}

void bfs(Graph\* graph, int startVertex) {

int queue[MAX\_VERTICES];

int front = 0, rear = -1;

visited[startVertex] = true;

printf("BFS Traversal: %d ", startVertex);

queue[++rear] = startVertex;

while (front <= rear) {

int currentVertex = queue[front++];

for (int i = 0; i < graph->numVertices; i++) {

if (graph->adjacencyMatrix[currentVertex][i] == 1 && !visited[i]) {

visited[i] = true;

printf("%d ", i);

queue[++rear] = i;

}

}

}

printf("\n");

}

void dfsRecursive(Graph\* graph, int startVertex) {

visited[startVertex] = true;

printf("%d ", startVertex);

for (int i = 0; i < graph->numVertices; i++) {

if (graph->adjacencyMatrix[startVertex][i] == 1 && !visited[i]) {

dfsRecursive(graph, i);

}

}

}

void dfs(Graph\* graph, int startVertex) {

for (int i = 0; i < graph->numVertices; i++) {

visited[i] = false;

}

printf("DFS Traversal (Recursive): ");

dfsRecursive(graph, startVertex);

printf("\n");

}

int main() {

Graph graph;

initializeGraph(&graph, 6);

addEdge(&graph, 0, 1);

addEdge(&graph, 0, 2);

addEdge(&graph, 1, 3);

addEdge(&graph, 1, 4);

addEdge(&graph, 2, 4);

addEdge(&graph, 3, 4);

addEdge(&graph, 3, 5);

addEdge(&graph, 4, 5);

int startVertex = 0;

bfs(&graph, startVertex);

dfs(&graph, startVertex);

return 0;

}