Graph Traversal

Write a C program to create a graph and perform a Breadth First Search and Depth First Search.

PROGRAM:

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

#define MAX 100

struct Graph {
    int numVertices;
    int adjMatrix[MAX][MAX];
    int visited[MAX];
};

void initGraph(struct Graph* graph, int vertices) {
    graph->numVertices = vertices;
    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            graph->adjMatrix[i][j] = 0;
        }
        graph->visited[i] = 0;
    }
}

void addEdge(struct Graph* graph, int src, int dest) {
    graph->adjMatrix[src][dest] = 1;
    graph->adjMatrix[dest][src] = 1;
}

void BFS(struct Graph* graph, int startVertex) {
    int queue[MAX], front = 0, rear = 0;
    for (int i = 0; i < graph->numVertices; i++) {
        graph->visited[i] = 0;
```

```
    }

    graph->visited[startVertex] = 1;
    queue[rear++] = startVertex;

    printf("BFS Traversal: ");
    while (front != rear) {
        int currentVertex = queue[front++];
        printf("%d ", currentVertex);

        for (int i = 0; i < graph->numVertices; i++) {
            if (graph->adjMatrix[currentVertex][i] == 1 && !graph->visited[i]) {
                queue[rear++] = i;
                graph->visited[i] = 1;
            }
        }
    }
    printf("\n");
}

void DFSUtil(struct Graph* graph, int vertex) {
    graph->visited[vertex] = 1;
    printf("%d ", vertex);

    for (int i = 0; i < graph->numVertices; i++) {
        if (graph->adjMatrix[vertex][i] == 1 && !graph->visited[i]) {
            DFSUtil(graph, i);
        }
    }
}

void DFS(struct Graph* graph, int startVertex) {
    for (int i = 0; i < graph->numVertices; i++) {
        graph->visited[i] = 0;
```

```
    }
    printf("DFS Traversal: ");
    DFSUtil(graph, startVertex);
    printf("\n");
}

int main() {
    struct Graph graph;
    int vertices = 6;

    initGraph(&graph, vertices);

    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);

    printf("Graph created with %d vertices.\n", vertices);

    BFS(&graph, 0);
    DFS(&graph, 0);

    return 0;
}
```

OUTPUT:

```
Graph created with 6 vertices.
BFS Traversal: 0 1 2 3 4 5
DFS Traversal: 0 1 3 4 2 5
```