Graph Traversal

Write a C program to create a graph and find a minimum spanning tree using prims algorithm.

PROGRAM:

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>

#define MAX 100

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

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

int minKey(int key[], bool mstSet[], int vertices) {
    int min = INT_MAX, minIndex;

    for (int v = 0; v < vertices; v++) {
        if (mstSet[v] == false && key[v] < min) {
```

```
            min = key[v];
            minIndex = v;
        }
    }
    return minIndex;
}

void printMST(int parent[], struct Graph* graph) {
    printf("Edge \tWeight\n");
    for (int i = 1; i < graph->numVertices; i++) {
        printf("%d - %d \t%d \n", parent[i], i, graph->adjMatrix[i][parent[i]]);
    }
}

void primMST(struct Graph* graph) {
    int vertices = graph->numVertices;
    int parent[MAX];
    int key[MAX];
    bool mstSet[MAX];

    for (int i = 0; i < vertices; i++) {
        key[i] = INT_MAX;
        mstSet[i] = false;
    }

    key[0] = 0;
    parent[0] = -1;

    for (int count = 0; count < vertices - 1; count++) {
        int u = minKey(key, mstSet, vertices);
        mstSet[u] = true;

        for (int v = 0; v < vertices; v++) {
            if (graph->adjMatrix[u][v] && mstSet[v] == false && graph->adjMatrix[u][v] < key[v]) {
```

```
            parent[v] = u;
            key[v] = graph->adjMatrix[u][v];
        }
    }
  }

  printMST(parent, graph);
}

int main() {
  struct Graph graph;
  int vertices = 5;

  initGraph(&graph, vertices);

  addEdge(&graph, 0, 1, 2);
  addEdge(&graph, 0, 3, 6);
  addEdge(&graph, 1, 2, 3);
  addEdge(&graph, 1, 3, 8);
  addEdge(&graph, 1, 4, 5);
  addEdge(&graph, 2, 4, 7);
  addEdge(&graph, 3, 4, 9);

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

  return 0;
}
```

OUTPUT:

```
Graph created with 5 vertices.
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
```