Prim's Algorithm

Aim:

The aim of the provided code is to implement Prim's algorithm in C to find the minimum spanning tree (MST) of a given graph.

Algorithm:

1. Start
2. Input the number of vertices and the adjacency matrix representing the graph.
3.Find the vertex with the minimum key value among the vertices not yet included in the MST.
4. Initializes key values and mstset for all vertices , then iteratively select the vertex with the minimum key value and updates the key values of its adjacent vertices of a shorter edge is found.
5.Print the edges of the MST along with their weights.
6. End.

Program:

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

#define MAX_VERTICES 10
#define INF 999999

int graph[MAX_VERTICES][MAX_VERTICES];
int vertices;

void createGraph() {
    int i, j;
    printf("Enter the number of vertices: ");
    scanf("%d", &vertices);
    printf("Enter the adjacency matrix:\n");
    for (i = 0; i < vertices; i++) {
        for (j = 0; j < vertices; j++) {
            scanf("%d", &graph[i][j]);
        }
    }
}

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

```
        }
    }
    return min_index;
}

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

void primMST() {
    int parent[vertices];
    int key[vertices];
    bool mstSet[vertices];

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

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

    for (int count = 0; count < vertices - 1; count++) {
        int u = findMinKey(key, mstSet);
        mstSet[u] = true;
        for (int v = 0; v < vertices; v++) {
            if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }

    printMST(parent);
}

int main() {
    createGraph();
    primMST();
    return 0;
}
```

Output:

```
Enter the number of vertices: 5
Enter the adjacency matrix:
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
0 5 7 9 0
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
```

Result:

The output is verified successfully for the above program.