NAME: SWETHA.J
REGISTER NO.:230701357

EX-13: Implementation of Prim's Algorithm

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

#define MAX_VERTICES 100

// Function to find the vertex with the minimum key value
int minKey(int key[], int mstSet[], int vertices) {
    int min = INT_MAX, minIndex;

    for (int v = 0; v < vertices; v++) {
        if (!mstSet[v] && key[v] < min) {
            min = key[v];
            minIndex = v;
        }
    }

    return minIndex;
}

// Function to print the constructed MST stored in parent[]
void printMST(int parent[], int graph[MAX_VERTICES][MAX_VERTICES], int
vertices) {
    printf("Edge \tWeight\n");
    for (int i = 1; i < vertices; i++) {
        printf("%d - %d \t%d\n", parent[i], i, graph[i][parent[i]]);
    }
}

// Function to implement Prim's algorithm for a given graph
void primMST(int graph[MAX_VERTICES][MAX_VERTICES], int vertices) {
    int parent[MAX_VERTICES]; // Array to store the constructed MST
    int key[MAX_VERTICES];    // Key values used to pick the minimum
weight edge
    int mstSet[MAX_VERTICES]; // To represent set of vertices included
in MST

    // Initialize all keys as INFINITE and mstSet[] as false
    for (int i = 0; i < vertices; i++) {
        key[i] = INT_MAX;
        mstSet[i] = 0;
    }

    // Always include the first vertex in the MST
    key[0] = 0; // Make key 0 so that this vertex is picked as the
first vertex
    parent[0] = -1; // First node is always the root of the MST

    // The MST will have vertices-1 edges
    for (int count = 0; count < vertices - 1; count++) {
        // Pick the minimum key vertex from the set of vertices not yet
included in the MST
        int u = minKey(key, mstSet, vertices);
```

```
        // Add the picked vertex to the MST Set
        mstSet[u] = 1;

        // Update key value and parent index of the adjacent vertices
        for (int v = 0; v < vertices; v++) {
            // graph[u][v] is non-zero only for adjacent vertices of m
            // mstSet[v] is false for vertices not yet included in MST
            // Update the key only if the graph[u][v] is smaller than
the key[v]
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }

    // Print the constructed MST
    printMST(parent, graph, vertices);
}

int main() {
    int vertices;

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

    if (vertices <= 0 || vertices > MAX_VERTICES) {
        printf("Invalid number of vertices. Exiting...\n");
        return 1;
    }

    int graph[MAX_VERTICES][MAX_VERTICES];

    // Input the adjacency matrix representing the graph
    printf("Input the adjacency matrix for the graph:\n");
    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            scanf("%d", &graph[i][j]);
        }
    }

    // Perform Prim's algorithm to find the MST
    primMST(graph, vertices);

    return 0;
}
```