WEEK 13 - Implementation of Prim’s Algorithm

```
#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: 2

Enter the adjacency matrix:

22

25

26

26

Edge Weight

0 - 1 26