LAB TASK –09

PRIM'S ALGORITHM CODE

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

#define V 5

int minKey(int key[], int mstSet[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
        if (mstSet[v] == 0 && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}

void primMST(int graph[V][V]) {
    int parent[V];
    int key[V];
    int mstSet[V];

    for (int i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = 0;

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

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

        for (int v = 0; v < V; v++)
            if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }

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

```c
int main() {
    int graph[V][V] = {
        {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},
    };

    primMST(graph);
    return 0;
}
```

OUTPUT

```
PS C:\Users\KIIT0001\DAA  LAB> cd "c:\Users\KIIT0001\DAA  LAB\" ; if ($?) { gcc prims.c -o prims } ; if ($?) { .\prims }
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
PS C:\Users\KIIT0001\DAA  LAB>
```

KRUSKAL'S ALGORITHM CODE

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

struct Edge {
    int src, dest, weight;
};



struct subset {
    int parent;
    int rank;
};



int find(struct subset subsets[], int i) {
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}
```

```c
void Union(struct subset subsets[], int x, int y) {
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);

    if (subsets[xroot].rank < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank > subsets[yroot].rank)
        subsets[yroot].parent = xroot;
    else {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}



int compare(const void* a, const void* b) {
    struct Edge* a1 = (struct Edge*)a;
    struct Edge* b1 = (struct Edge*)b;
    return a1->weight - b1->weight;
}

void KruskalMST(struct Edge edges[], int V, int E) {
    struct Edge result[V];
    int e = 0;
    int i = 0;

    qsort(edges, E, sizeof(struct Edge), compare);

    struct subset* subsets = (struct subset*)malloc(V * sizeof(struct subset));
    for (int v = 0; v < V; ++v) {
        subsets[v].parent = v;
        subsets[v].rank = 0;
    }

    while (e < V - 1 && i < E) {
        struct Edge next = edges[i++];

        int x = find(subsets, next.src);
        int y = find(subsets, next.dest);
```

```
        if (x != y) {
            result[e++] = next;
            Union(subsets, x, y);
        }
    }

    printf("Edge \tWeight\n");
    for (i = 0; i < e; ++i)
        printf("%d - %d \t%d\n", result[i].src, result[i].dest, result[i].weight);

    free(subsets);
}

int main() {
    int V = 5;
    int E = 7;
    struct Edge edges[] = {
        {0, 1, 2}, {0, 3, 6}, {1, 2, 3},

        {1, 3, 8}, {1, 4, 5}, {2, 4, 7}, {3, 4, 9}
    };

    KruskalMST(edges, V, E);
    return 0;
}
```

OUTPUT

```
PS C:\Users\KIIT0001\DAA  LAB> cd "c:\Users\KIIT0001\DAA  LAB\" ; if ($?) { gcc kruskal.c -o kruskal } ; if ($?) { .\kruskal }
Edge    Weight
0 - 1   2
1 - 2   3
1 - 4   5
0 - 3   6
PS C:\Users\KIIT0001\DAA  LAB>
```