DAA

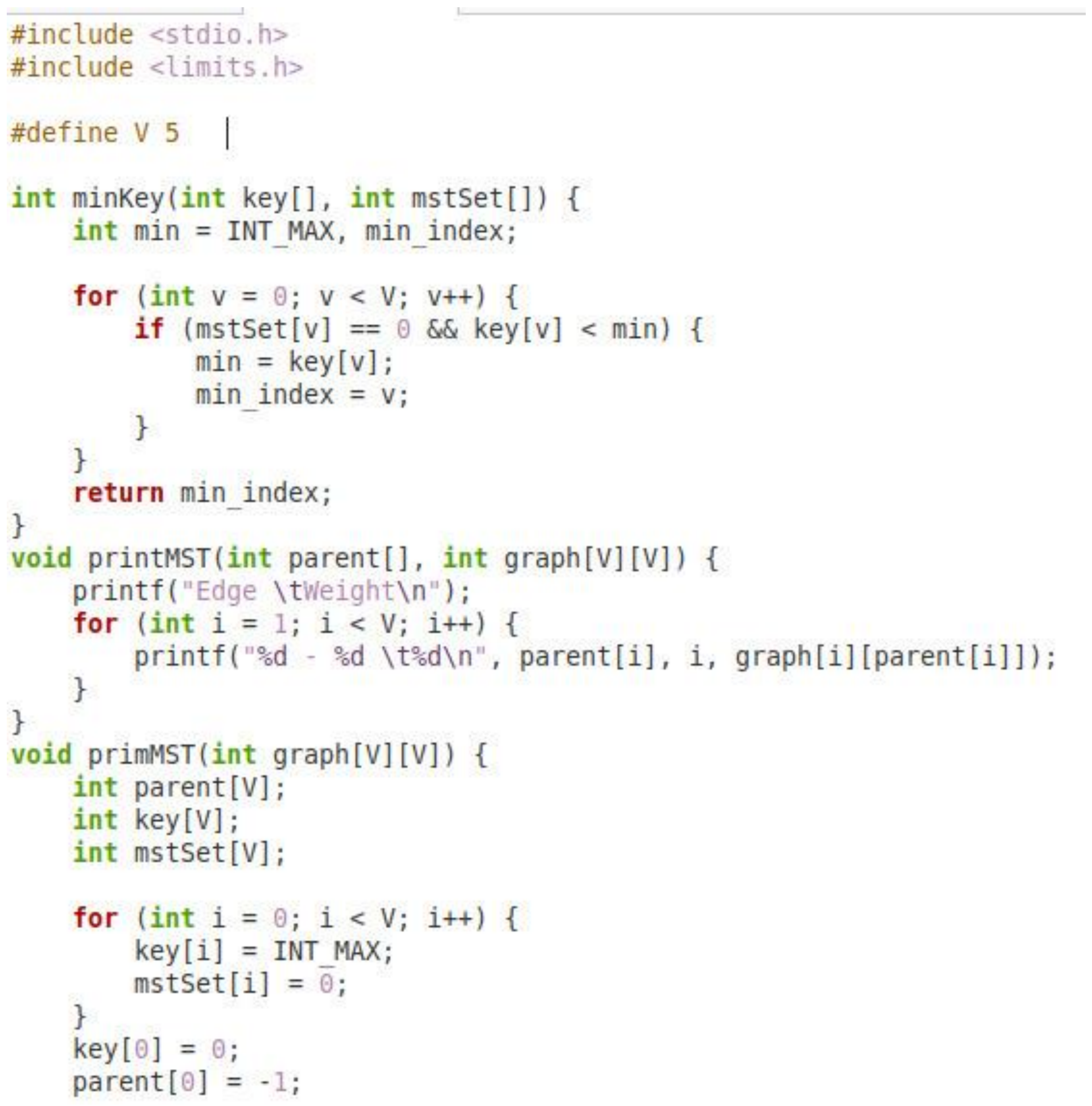
WEEK-4

NAME: KANTAMRAJU SREEJA

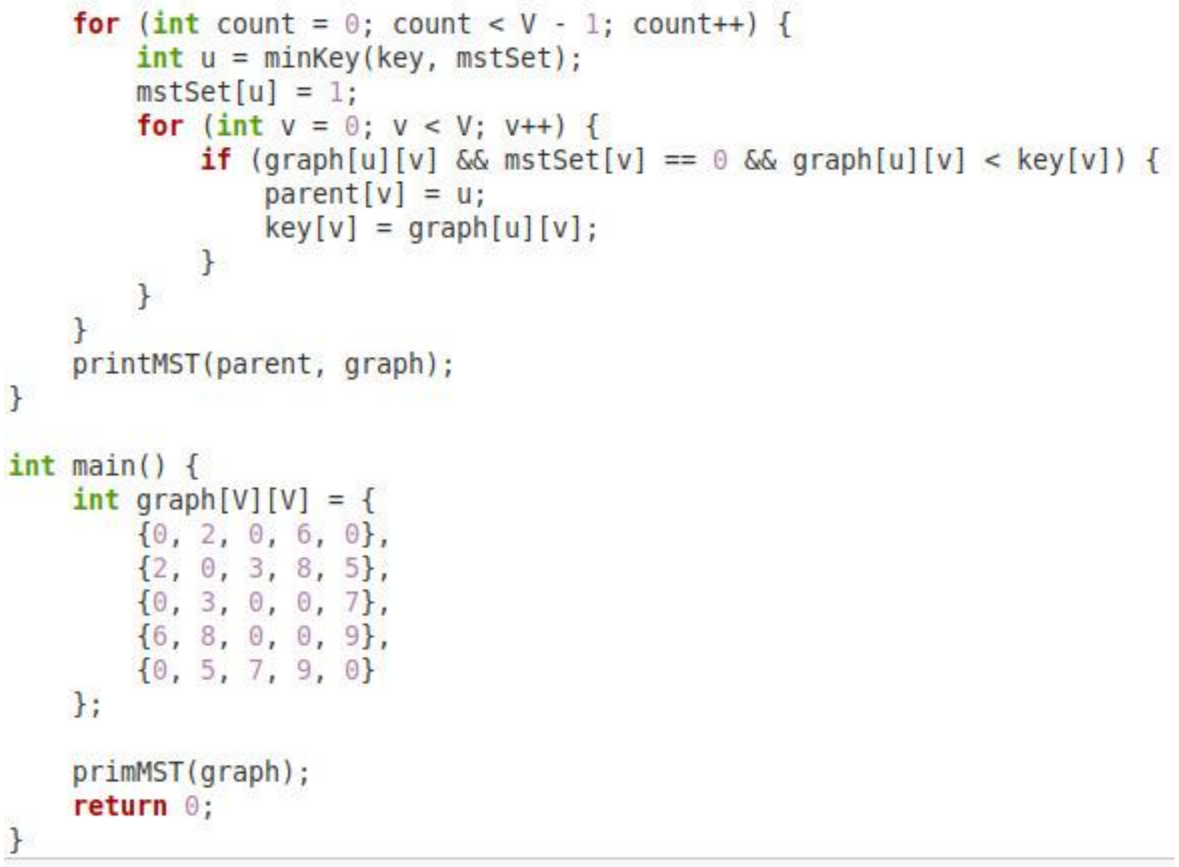
ROLL NO: CH.SC.U4CSE24255

1)PRIMS ALGORITHM:

CODE:

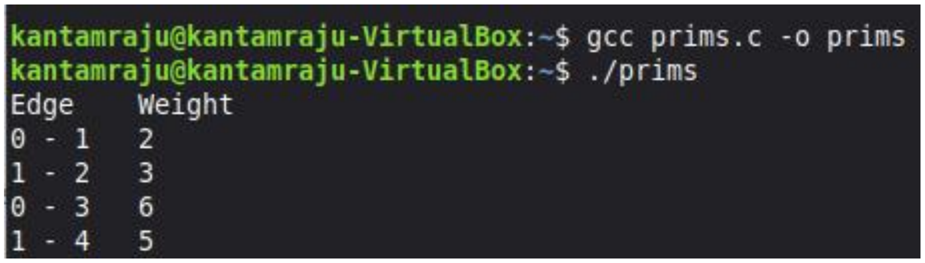


```c
#include <stdio.h>
#include <limits.h>

#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 printMST(int parent[], int graph[V][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]]);
    }
}
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;
```



```c
    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];
            }
        }
    }
    printMST(parent, graph);
}

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:



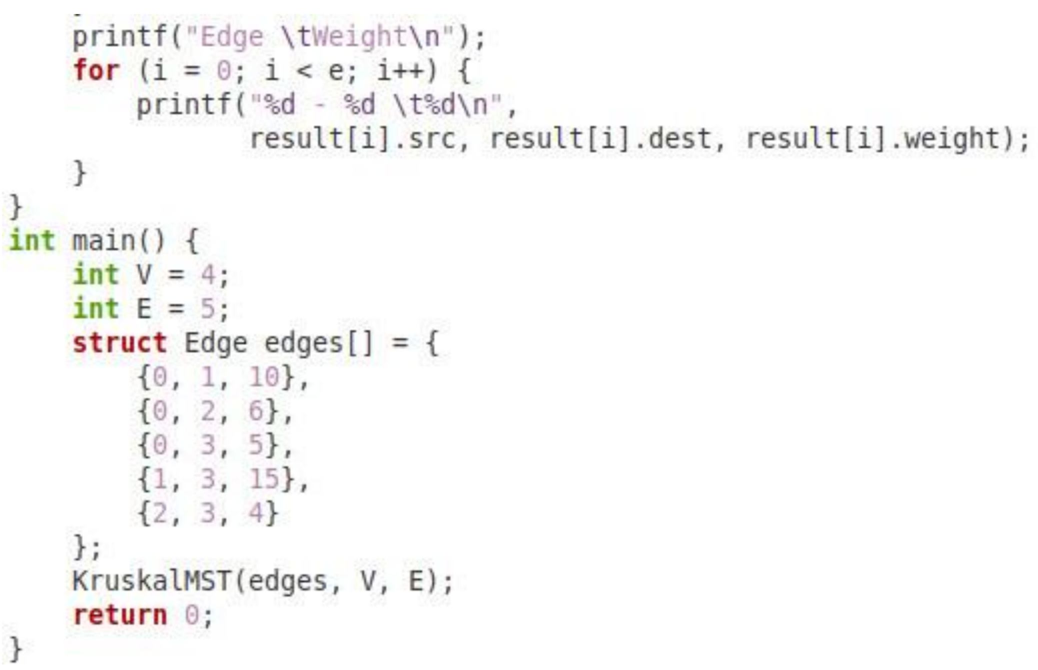
```
kantamraju@kantamraju-VirtualBox:~$ gcc prims.c -o prims
kantamraju@kantamraju-VirtualBox:~$ ./prims
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
```

2) KRUSHALS ALGORITHM:

CODE:

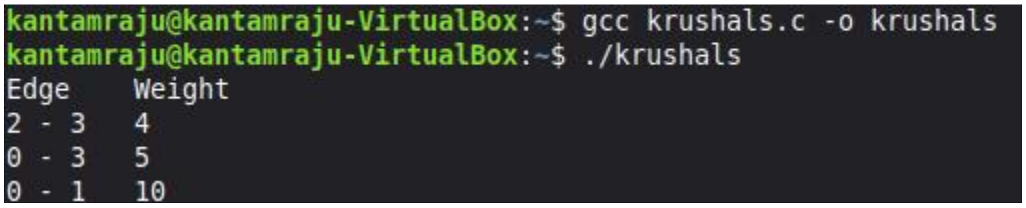
```c
#include <stdio.h>
struct Edge {
    int src, dest, weight;
};
struct Subset {
    int parent;
    int rank;
};
int find(struct Subset subsets[], int i) {
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}
void Union(struct 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++;
    }
}
```

```c
void sortEdges(struct Edge edges[], int E) {
    struct Edge temp;
    for (int i = 0; i < E - 1; i++) {
        for (int j = 0; j < E - i - 1; j++) {
            if (edges[j].weight > edges[j + 1].weight) {
                temp = edges[j];
                edges[j] = edges[j + 1];
                edges[j + 1] = temp;
            }
        }
    }
}

// Kruskal's Algorithm
void KruskalMST(struct Edge edges[], int V, int E) {
    struct Edge result[V];
    struct Subset subsets[V];

    for (int v = 0; v < V; v++) {
        subsets[v].parent = v;
        subsets[v].rank = 0;
    }
    sortEdges(edges, E);
    int e = 0, i = 0;
    while (e < V - 1 && i < E) {
        struct Edge next_edge = edges[i++];
        int x = find(subsets, next_edge.src);
        int y = find(subsets, next_edge.dest);
        if (x != y) {
            result[e++] = next_edge;
            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);
    }
}
int main() {
    int V = 4;
    int E = 5;
    struct Edge edges[] = {
        {0, 1, 10},
        {0, 2, 6},
        {0, 3, 5},
        {1, 3, 15},
        {2, 3, 4}
    };
    KruskalMST(edges, V, E);
    return 0;
}
```

OUTPUT:



```
kantamraju@kantamraju-VirtualBox:~$ gcc krushals.c -o krushals
kantamraju@kantamraju-VirtualBox:~$ ./krushals
Edge    Weight
2 - 3   4
0 - 3   5
0 - 1   10
```