DAA ASSIGNMENT 4

Prim's algorithm:

Code:

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

int minKey(int n, int key[], int mstSet[]) {
   int min = INT_MAX, min_index = -1;
   for (int v = 0; v < n; v++) {
      if (!mstSet[v] && key[v] < min) {
         min = key[v];
         min_index = v;
      }
   }
   return min_index;
}

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

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

   for (int i = 0; i < n; i++) {
      key[i] = INT_MAX;
      mstSet[i] = 0;
   }
```

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

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

    printMST(n, parent, graph);
}

int main() {
    int n;
    scanf("%d", &n);
    int graph[n][n];

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &graph[i][j]);
        }
    }

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

Output:

```
pooja@DESKTOP-5DBTBML:~$ nano primalo.c
pooja@DESKTOP-5DBTBML:~$ gcc primalo.c -o primalo
pooja@DESKTOP-5DBTBML:~$ ./primalo
3
2
0
5
4
6
3
0
8
5
Edge Weight
2 - 1 3
0 - 2 0
```

Kruskal's algorithum:

Code:

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

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

int parent[10];
int find(int i) {
   while (parent[i] != i)
      i = parent[i];
   return i;
}

int uni(int i, int j) {
   if (i != j) {
      parent[j] = i;
      return 1;
   }
```

```
    return 0;
}

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

int main() {
    int n, m, i, mincost = 0, edges_used = 0;
    struct Edge edges[20];

    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter number of edges: ");
    scanf("%d", &m);

    for (i = 0; i < m; i++) {
        printf("Edge %d - Source Destination Weight: ", i+1);
        scanf("%d %d %d", &edges[i].src, &edges[i].dest, &edges[i].weight);
    }

    for (i = 0; i < n; i++) parent[i] = i;

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

    printf("\nMST Edges:\n");
    for (i = 0; i < m && edges_used < n-1; i++) {
        int u = find(edges[i].src-1);
        int v = find(edges[i].dest-1);
        if (uni(u, v)) {
            printf("Edge (%d,%d) weight = %d\n",
                  edges[i].src, edges[i].dest, edges[i].weight);
            mincost += edges[i].weight;
            edges_used++;
        }
    }
```

```
    printf("\nTotal Minimum Weight = %d\n", mincost);
    return 0;
}
```

Output:

```
pooja@DESKTOP-5DBTBML:~$ nano kruskalalo.c
pooja@DESKTOP-5DBTBML:~$ gcc kruskalalo.c -o kruskalalo
pooja@DESKTOP-5DBTBML:~$ ./kruskalalo
Enter number of vertices: 4
Enter number of edges: 5
Edge 1 - Source Destination Weight: 0 1 10
Edge 2 - Source Destination Weight: 2 3 5
Edge 3 - Source Destination Weight: 4 3 6
Edge 4 - Source Destination Weight: 7 6 8
Edge 5 - Source Destination Weight: 4 0 8

MST Edges:
Edge (2,3) weight = 5
Edge (4,3) weight = 6
Edge (4,0) weight = 8
```