1. **Kruskal**

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_EDGES 100

typedef struct {
    int u, v, weight;
} Edge;

int parent[100];

int find(int i) {
    if (parent[i] == i)
        return i;
    return parent[i] = find(parent[i]);
}

void union_set(int u, int v) {
    int xset = find(u);
    int yset = find(v);
    parent[xset] = yset;
}

int compare(const void *a, const void *b) {
    Edge *e1 = (Edge *)a;
    Edge *e2 = (Edge *)b;
    return e1->weight - e2->weight;
}

void kruskal(Edge edges[], int n, int E) {
    int i, count = 0, total_weight = 0;
    for (i = 0; i < n; i++)
        parent[i] = i;
    qsort(edges, E, sizeof(Edge), compare);
    printf("Edges in MST:\n");
    for (i = 0; i < E && count < n - 1; i++) {
        int u_set = find(edges[i].u);
        int v_set = find(edges[i].v);
        if (u_set != v_set) {
            printf("%d - %d : %d\n", edges[i].u, edges[i].v, edges[i].weight);
```

```
            total_weight += edges[i].weight;
            union_set(u_set, v_set);
            count++;
        }
    }
    printf("Total weight of MST: %d\n", total_weight);
}

int main() {
    int n, E;
    printf("Enter number of vertices and edges: ");
    scanf("%d %d", &n, &E);
    Edge edges[MAX_EDGES];
    printf("Enter edges (u v weight):\n");
    for (int i = 0; i < E; i++)
        scanf("%d %d %d", &edges[i].u, &edges[i].v, &edges[i].weight);
    kruskal(edges, n, E);
    return 0;
}
```

```
Enter number of vertices and edges: 7 9
Enter edges (u v weight):
6 1 10
3 4 12
7 2 14
2 3 16
7 4 18
5 4 22
5 7 24
5 6 25
1 2 28
Edges in MST:
6 - 1 : 10
3 - 4 : 12
7 - 2 : 14
2 - 3 : 16
5 - 4 : 22
5 - 6 : 25
Total weight of MST: 99


=== Code Execution Successful ===
```

2. Prim’s

```
#include <stdio.h>
#include <limits.h>
#define MAX 100

int n;
int graph[MAX][MAX];

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

void prim() {
    int parent[MAX];
    int key[MAX];
    int mstSet[MAX];
    int i, count, u, v;

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

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

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

        for (v = 0; v < n; 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 (i = 1; i < n; i++)
        printf("%d - %d \t%d\n", parent[i], i, graph[i][parent[i]]);
}

int main() {
    int i, j;
    printf("Enter number of vertices: ");
    scanf("%d", &n);

    printf("Enter the adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);

    prim();
    return 0;
}
```

```
Enter number of vertices: 4
Enter the adjacency matrix:
0 10 6 5
10 0 0 15
6 0 0 4
5 15 4 0
Edge    Weight
0 - 1   10
3 - 2   4
0 - 3   5


=== Code Execution Successful ===
```