

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

```
typedef struct edge {  
    int u, v, w;  
} edge;
```

```
typedef struct edge_list {  
    edge data[MAX];  
    int n;  
} edge_list;
```

```
edge_list elist;
```

```
int Graph[MAX][MAX], n;  
edge_list spanlist;
```

```
void kruskalAlgo();  
int find(int belongs[], int vertexno);  
void applyUnion(int belongs[], int c1, int c2);  
void sort();  
void print();
```

```
void kruskalAlgo() {  
    int belongs[MAX], i, j, cno1, cno2;  
    elist.n = 0;
```

```
    for (i = 1; i < n; i++)  
        for (j = 0; j < i; j++) {  
            if (Graph[i][j] != 0) {  
                elist.data[elist.n].u = i;
```

```
    elist.data[elist.n].v = j;

    elist.data[elist.n].w = Graph[i][j];

    elist.n++;
}
}
```

```
sort();
```

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

```
spanlist.n = 0;
```

```
for (i = 0; i < elist.n; i++) {
    cno1 = find(belongs, elist.data[i].u);
    cno2 = find(belongs, elist.data[i].v);

    if (cno1 != cno2) {
        spanlist.data[spanlist.n] = elist.data[i];
        spanlist.n = spanlist.n + 1;
        applyUnion(belongs, cno1, cno2);
    }
}
}
```

```
int find(int belongs[], int vertexno) {
    return (belongs[vertexno]);
}
```

```
void applyUnion(int belongs[], int c1, int c2) {
    int i;
```

```

for (i = 0; i < n; i++)
    if (belongs[i] == c2)
        belongs[i] = c1;
}

```

```

void sort() {
    int i, j;
    edge temp;

```

```

    for (i = 1; i < elist.n; i++)
        for (j = 0; j < elist.n - 1; j++)
            if (elist.data[j].w > elist.data[j + 1].w) {
                temp = elist.data[j];
                elist.data[j] = elist.data[j + 1];
                elist.data[j + 1] = temp;
            }
}

```

```

void print() {
    int i, cost = 0;

```

```

    for (i = 0; i < spanlist.n; i++) {
        printf("\n%d - %d : %d", spanlist.data[i].u, spanlist.data[i].v, spanlist.data[i].w);
        cost = cost + spanlist.data[i].w;
    }

```

```

    printf("\nSpanning tree cost: %d", cost);
}

```

```

int main() {

```

```
int i, j, total_cost;
```

```
n = 6;
```

```
Graph[0][0] = 0;
```

```
Graph[0][1] = 4;
```

```
Graph[0][2] = 4;
```

```
Graph[0][3] = 0;
```

```
Graph[0][4] = 0;
```

```
Graph[0][5] = 0;
```

```
Graph[0][6] = 0;
```

```
Graph[1][0] = 4;
```

```
Graph[1][1] = 0;
```

```
Graph[1][2] = 2;
```

```
Graph[1][3] = 0;
```

```
Graph[1][4] = 0;
```

```
Graph[1][5] = 0;
```

```
Graph[1][6] = 0;
```

```
Graph[2][0] = 4;
```

```
Graph[2][1] = 2;
```

```
Graph[2][2] = 0;
```

```
Graph[2][3] = 3;
```

```
Graph[2][4] = 4;
```

```
Graph[2][5] = 0;
```

```
Graph[2][6] = 0;
```

```
Graph[3][0] = 0;
```

```
Graph[3][1] = 0;
```

```
Graph[3][2] = 3;
```

```
Graph[3][3] = 0;
```

```
Graph[3][4] = 3;
```

```
Graph[3][5] = 0;
```

```
Graph[3][6] = 0;
```

```
Graph[4][0] = 0;
```

```
Graph[4][1] = 0;
```

```
Graph[4][2] = 4;
```

```
Graph[4][3] = 3;
```

```
Graph[4][4] = 0;
```

```
Graph[4][5] = 0;
```

```
Graph[4][6] = 0;
```

```
Graph[5][0] = 0;
```

```
Graph[5][1] = 0;
```

```
Graph[5][2] = 2;
```

```
Graph[5][3] = 0;
```

```
Graph[5][4] = 3;
```

```
Graph[5][5] = 0;
```

```
Graph[5][6] = 0;
```

```
kruskalAlgo();
```

```
print();
```

```
}
```

```
C:\Users\nagir\OneDrive\Documents\exp 25.exe

2 - 1 : 2
5 - 2 : 2
13 - 2 : 3
14 - 3 : 3
11 - 0 : 4
Spanning tree cost: 14
-----
Process exited after 0.05029 seconds with return value 23
Press any key to continue . . .
```