```c
/* 천인국, 공용해, 하상호 지음, C언어로 쉽게 풀어쓴 자료구조(개정3판), 생능출판사 프로그램 11.8 */

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

#define TRUE 1
#define FALSE 0

#define MAX_VERTICES 100
#define INF 1000

int parent[MAX_VERTICES];    // 부모 노드
                             // 초기화
void set_init(int n)
{
    for (int i = 0; i < n; i++)
        parent[i] = -1;
}
// curr가 속하는 집합을 반환한다.
int set_find(int curr)
{
    if (parent[curr] == -1)
        return curr;             // 루트
    while (parent[curr] != -1) curr = parent[curr];
    return curr;
}

// 두개의 원소가 속한 집합을 합친다.
void set_union(int a, int b)
{
    int root1 = set_find(a); // 노드 a의 루트를 찾는다.
    int root2 = set_find(b); // 노드 b의 루트를 찾는다.
    if (root1 != root2)  // 합한다.
        parent[root1] = root2;
}

struct Edge {            // 간선을 나타내는 구조체
    int start, end, weight;
};

typedef struct GraphType {
    int n;   // 간선의 개수
    struct Edge edges[2 * MAX_VERTICES];
} GraphType;

// 그래프 초기화
void graph_init(GraphType* g)
{
    g->n = 0;
    for (int i = 0; i < 2 * MAX_VERTICES; i++) {
        g->edges[i].start = 0;
        g->edges[i].end = 0;
        g->edges[i].weight = INF;
    }
}
// 간선 삽입 연산
void insert_edge(GraphType* g, int start, int end, int w)
{
    g->edges[g->n].start = start;
    g->edges[g->n].end = end;
    g->edges[g->n].weight = w;
    g->n++;
}
// qsort()에 사용되는 함수
int compare(const void* a, const void* b)
{
    struct Edge* x = (struct Edge*)a;
    struct Edge* y = (struct Edge*)b;
    return (x->weight - y->weight);
}

// kruskal의 최소 비용 신장 트리 프로그램
void kruskal(GraphType* g)
{
    int edge_accepted = 0;   // 현재까지 선택된 간선의 수
    int uset, vset;          // 정점 u와 정점 v의 집합 번호
    struct Edge e;

    set_init(g->n);              // 집합 초기화
    qsort(g->edges, g->n, sizeof(struct Edge), compare);

    printf("크루스칼 최소 신장 트리 알고리즘 \n");
    int i = 0;
    while (edge_accepted < (g->n - 1))    // 간선의 수 < (n-1)
    {
        e = g->edges[i];
        uset = set_find(e.start);        // 정점 u의 집합 번호
        vset = set_find(e.end);      // 정점 v의 집합 번호
        if (uset != vset) {          // 서로 속한 집합이 다르면
            printf("간선 (%d,%d) %d 선택\n", e.start, e.end, e.weight);
            edge_accepted++;
            set_union(uset, vset);   // 두개의 집합을 합친다.
        }
        i++;
    }
}
int main(void)
{
    GraphType* g;
    g = (GraphType*)malloc(sizeof(GraphType));
    graph_init(g);

    insert_edge(g, 0, 1, 29);
    insert_edge(g, 1, 2, 16);
    insert_edge(g, 2, 3, 12);
    insert_edge(g, 3, 4, 22);
    insert_edge(g, 4, 5, 27);
    insert_edge(g, 5, 0, 10);
    insert_edge(g, 6, 1, 15);
    insert_edge(g, 6, 3, 18);
    insert_edge(g, 6, 4, 25);

    kruskal(g);
    free(g);
    return 0;
}
```