**KRUSKALS ALGORITHM**

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

typedef struct {

int u, v, w;

} Edge;

Edge edges[MAX];

int parent[MAX], n, e;

int find(int i) {

while (parent[i] != i)

i = parent[i];

return i;

}

void union\_set(int i, int j) {

int a = find(i);

int b = find(j);

parent[a] = b;

}

int cmp(const void \*a, const void \*b) {

return ((Edge \*)a)->w - ((Edge \*)b)->w;

}

void kruskal() {

int i, count = 0, cost = 0;

for (i = 0; i < n; i++)

parent[i] = i;

qsort(edges, e, sizeof(Edge), cmp);

for (i = 0; i < e && count < n - 1; i++) {

int u = edges[i].u;

int v = edges[i].v;

if (find(u) != find(v)) {

printf("Edge: %d - %d (%d)\n", u, v, edges[i].w);

cost += edges[i].w;

union\_set(u, v);

count++;

}

}

printf("Total cost: %d\n", cost);

}

int main() {

printf("Enter number of vertices and edges: ");

scanf("%d %d", &n, &e);

printf("Enter each edge (u v weight):\n");

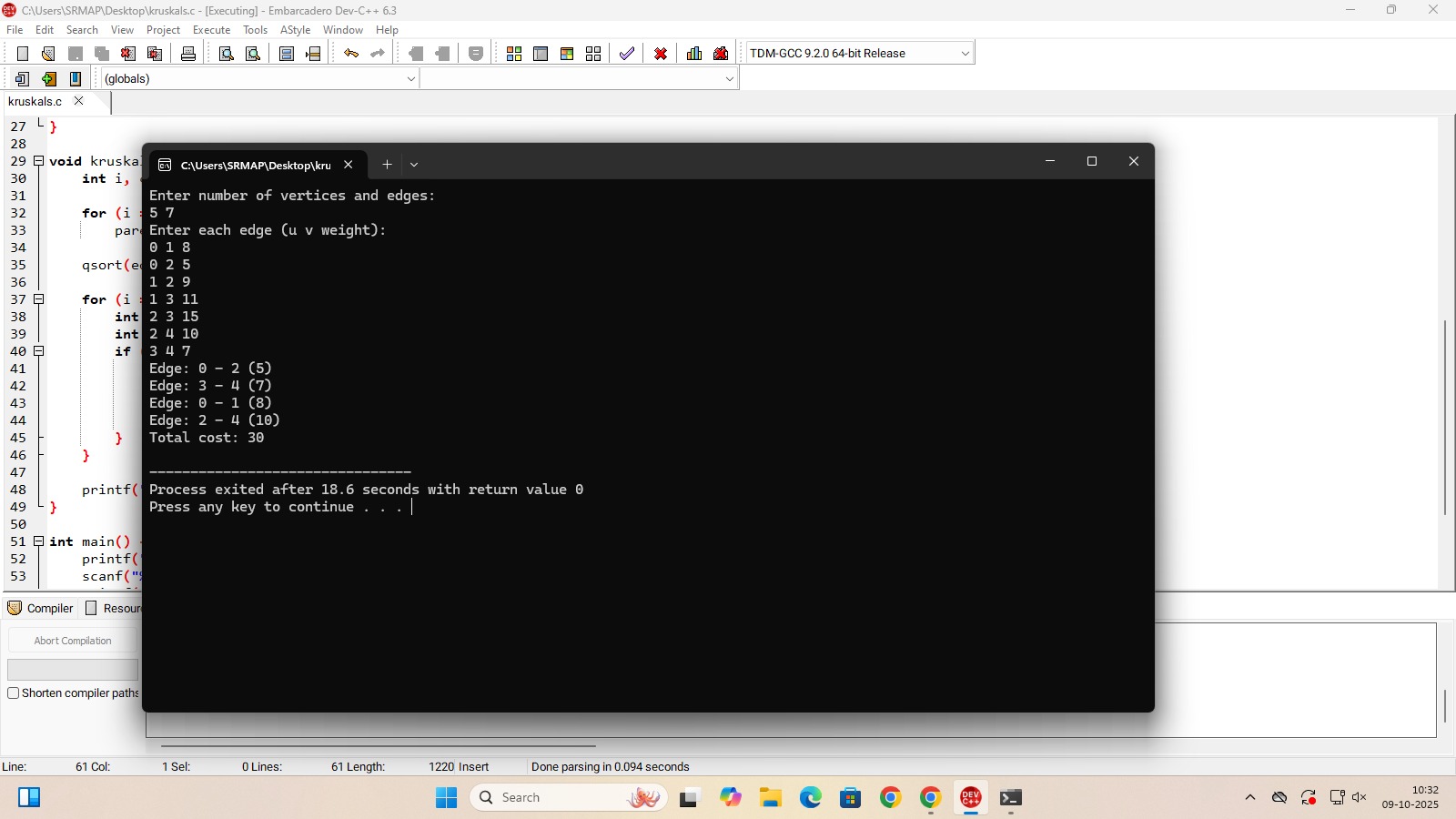
for (int i = 0; i < e; i++)

scanf("%d %d %d", &edges[i].u, &edges[i].v, &edges[i].w);

kruskal();

return 0;

}



**PRIMS ALGORITHM**

#include <stdio.h>

#include <limits.h>

#define MAX 100

#define INF 999999

Int cost[MAX][MAX], visited[MAX], n;

Void prim() {

Int edge\_count = 0, total\_cost = 0;

Visited[0] = 1;

While (edge\_count < n – 1) {

Int min = INF, a = -1, b = -1;

For (int I = 0; I < n; i++) {

If (visited[i]) {

For (int j = 0; j < n; j++) {

If (!visited[j] && cost[i][j] < min) {

Min = cost[i][j];

A = I;

B = j;

}

}

}

}

If (a != -1 && b != -1) {

Printf(“Edge: %d - %d (%d)\n”, a, b, cost[a][b]);

Visited[b] = 1;

Total\_cost += cost[a][b];

Edge\_count++;

}

}

Printf(“Total cost: %d\n”, total\_cost);

}

Int main() {

Printf(“Enter number of vertices: “);

Scanf(“%d”, &n);

Printf(“Enter adjacency matrix (use 999 for no edge):\n”);

For (int I = 0; I < n; i++)

For (int j = 0; j < n; j++)

Scanf(“%d”, &cost[i][j]);

For (int I = 0; I < n; i++)

Visited[i] = 0;

Prim();

Return 0;

}

