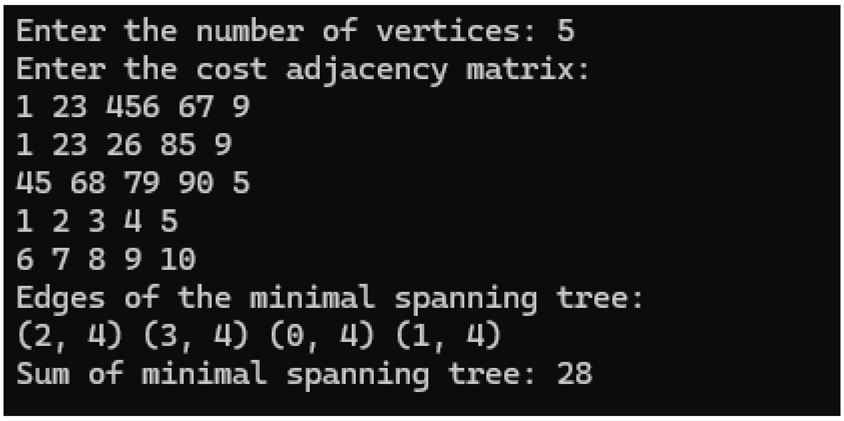
KRUSKALS

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

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

struct Graph {
    int V, E;
    struct Edge* edge;
};

struct Subset {
    int parent;
    int rank;
};

struct Graph* createGraph(int V, int E);
int find(struct Subset subsets[], int i);
void Union(struct Subset subsets[], int x, int y);
void KruskalMST(struct Graph* graph);
void printMST(struct Edge result[], int e, int totalWeight);

int main() {
    int V, E = 0;
    int i, j, weight;

    printf("Enter the number of vertices: ");
    scanf("%d", &V);

    int adjMatrix[V][V];

    printf("Enter the cost adjacency matrix:\n");
    for (i = 0; i < V; i++) {
        for (j = 0; j < V; j++) {
            scanf("%d", &adjMatrix[i][j]);
            if (i < j && adjMatrix[i][j] != 0) {
                E++;
            }
        }
    }

    struct Graph* graph = createGraph(V, E);

    int e = 0;
    for (i = 0; i < V; i++) {
        for (j = i+1; j < V; j++) {
            if (adjMatrix[i][j] != 0) {
                graph->edge[e].src = i;
                graph->edge[e].dest = j;
                graph->edge[e].weight = adjMatrix[i][j];
                e++;
            }
        }
    }

    KruskalMST(graph);

    free(graph->edge);
    free(graph);

    return 0;
}

struct Graph* createGraph(int V, int E) {
    struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
    if (!graph) {
        printf("Memory allocation failed\n");
        exit(1);
    }

    graph->V = V;
    graph->E = E;

    graph->edge = (struct Edge*)malloc(E * sizeof(struct Edge));
    if (!graph->edge) {
        printf("Memory allocation failed\n");
        exit(1);
    }

    return graph;
}

int find(struct Subset subsets[], int i) {
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);

    return subsets[i].parent;
}

void Union(struct Subset subsets[], int x, int y) {
    int rootX = find(subsets, x);
    int rootY = find(subsets, y);

    if (subsets[rootX].rank < subsets[rootY].rank)
        subsets[rootX].parent = rootY;
    else if (subsets[rootX].rank > subsets[rootY].rank)
```

```
        subsets[rootY].parent = rootX;
    else {
        subsets[rootY].parent = rootX;
        subsets[rootX].rank++;
    }
}

int compareEdges(const void* a, const void* b) {
    struct Edge* a1 = (struct Edge*)a;
    struct Edge* b1 = (struct Edge*)b;
    return a1->weight - b1->weight;
}

void KruskalMST(struct Graph* graph) {
    int V = graph->V;
    struct Edge result[V-1];
    int e = 0;
    int i = 0;
    int totalWeight = 0;

    qsort(graph->edge, graph->E, sizeof(graph->edge[0]), compareEdges);

    struct Subset* subsets = (struct Subset*)malloc(V * sizeof(struct Subset));
    if (!subsets) {
        printf("Memory allocation failed\n");
        exit(1);
    }

    for (int v = 0; v < V; v++) {
        subsets[v].parent = v;
        subsets[v].rank = 0;
    }

    while (e < V - 1 && i < graph->E) {
        struct Edge next_edge = graph->edge[i++];

        int x = find(subsets, next_edge.src);
        int y = find(subsets, next_edge.dest);

        if (x != y) {
            result[e++] = next_edge;
            totalWeight += next_edge.weight;
            Union(subsets, x, y);
        }
    }

    if (e != V - 1) {
        printf("Graph is not connected. MST not possible.\n");
    } else {
        printMST(result, e, totalWeight);
    }

    free(subsets);
}

void printMST(struct Edge result[], int e, int totalWeight) {
    printf("Edges of the minimal spanning tree:\n");
    for (int i = 0; i < e; i++) {
        printf("(%d, %d) ", result[i].src, result[i].dest);
    }
    printf("\nSum of minimal spanning tree: %d\n", totalWeight);
}
```

OUTPUT:



```
Enter the number of vertices: 5
Enter the cost adjacency matrix:
1 23 456 67 9
1 23 26 85 9
45 68 79 90 5
1 2 3 4 5
6 7 8 9 10
Edges of the minimal spanning tree:
(2, 4) (3, 4) (0, 4) (1, 4)
Sum of minimal spanning tree: 28
```