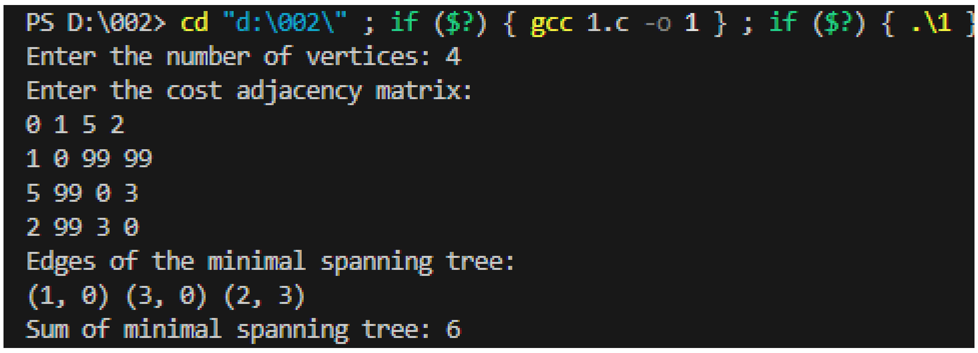
LAB-4

(i) Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.

```c
//prim's algorithm
#include <stdio.h>
int cost[10][10], n, t[10][2], sum;
void prims(int cost[10][10], int n);
int main() {
    int i, j;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the cost adjacency matrix:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &cost[i][j]);
        }
    }
    prims(cost, n);
    printf("Edges of the minimal spanning tree:\n");
    for (i = 0; i < n - 1; i++) {
        printf("(%d, %d) ", t[i][0], t[i][1]);
    }
    printf("\nSum of minimal spanning tree: %d\n", sum);
    return 0;
}
void prims(int cost[10][10], int n) {
    int i, j, u, v;
    int min, source;
    int p[10], d[10], s[10];
    min = 999;
    source = 0;
    for (i = 0; i < n; i++) {
        d[i] = cost[source][i];
        s[i] = 0;
        p[i] = source;
    }
    s[source] = 1;
    sum = 0;
    int k = 0;
    for (i = 0; i < n - 1; i++) {
        min = 999;
        u = -1;
        for (j = 0; j < n; j++) {
            if (s[j] == 0 && d[j] < min) {
                min = d[j];
                u = j;
            }
        }
        if (u != -1) {
            t[k][0] = u;
            t[k][1] = p[u];
            k++;
            sum += cost[u][p[u]];
            s[u] = 1;
            for (v = 0; v < n; v++) {
                if (s[v] == 0 && cost[u][v] < d[v]) {
                    d[v] = cost[u][v];
                    p[v] = u;
                }
            }
        }
    }
}
```

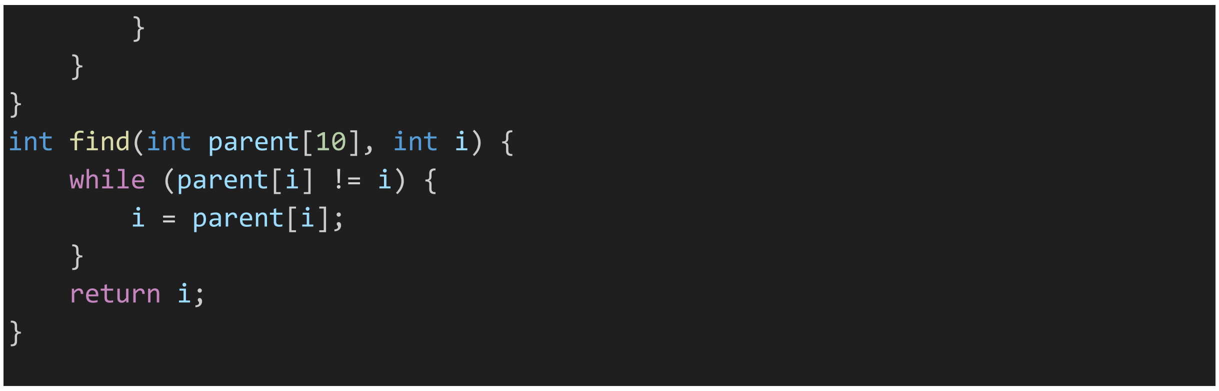
OUTPUT:



```
PS D:\002> cd "d:\002\" ; if ($?) { gcc 1.c -o 1 } ; if ($?) { .\1 }
Enter the number of vertices: 4
Enter the cost adjacency matrix:
0 1 5 2
1 0 99 99
5 99 0 3
2 99 3 0
Edges of the minimal spanning tree:
(1, 0) (3, 0) (2, 3)
Sum of minimal spanning tree: 6
```

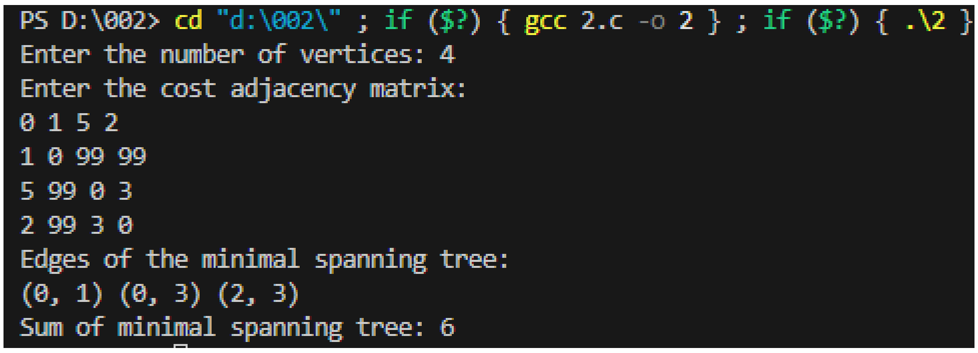
(ii) Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal’s algorithm.

```c
//Kruskal’s algorithm
#include <stdio.h>
int cost[10][10], n, t[10][2], sum;
void kruskal(int cost[10][10], int n);
int find(int parent[10], int i);
int main() {
    int i, j;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the cost adjacency matrix:\n");
    for (i = 0; i < n; i++) {
         for (j = 0; j < n; j++) {
             scanf("%d", &cost[i][j]);
        }
    }
    kruskal(cost, n);
    printf("Edges of the minimal spanning tree:\n");
    for (i = 0; i < n - 1; i++) {
        printf("(%d, %d) ", t[i][0], t[i][1]);
    }
    printf("\nSum of minimal spanning tree: %d\n", sum);
    return 0;
}
void kruskal(int cost[10][10], int n) {

    int min, u, v, count, k;
    int parent[10];
    k = 0;
    sum = 0;
    for (int i = 0; i < n; i++) {
        parent[i] = i;
    }
    count = 0;
    while (count < n - 1) {
        min = 999;
        u = -1;
        v = -1;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (find(parent, i) != find(parent, j) && cost[i][j] < min) {
                    min = cost[i][j];
                    u = i;
                    v = j;
                }
            }
        }

        int root_u = find(parent, u);
        int root_v = find(parent, v);
        if (root_u != root_v) {
            parent[root_u] = root_v;
            t[k][0] = u;
            t[k][1] = v;
            sum += min;
            k++;
            count++;
```



```c
        }
    }
}
int find(int parent[10], int i) {
    while (parent[i] != i) {
        i = parent[i];
    }
    return i;
}
```

OUTPUT:



```
PS D:\002> cd "d:\002\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Enter the number of vertices: 4
Enter the cost adjacency matrix:
0 1 5 2
1 0 99 99
5 99 0 3
2 99 3 0
Edges of the minimal spanning tree:
(0, 1) (0, 3) (2, 3)
Sum of minimal spanning tree: 6
```