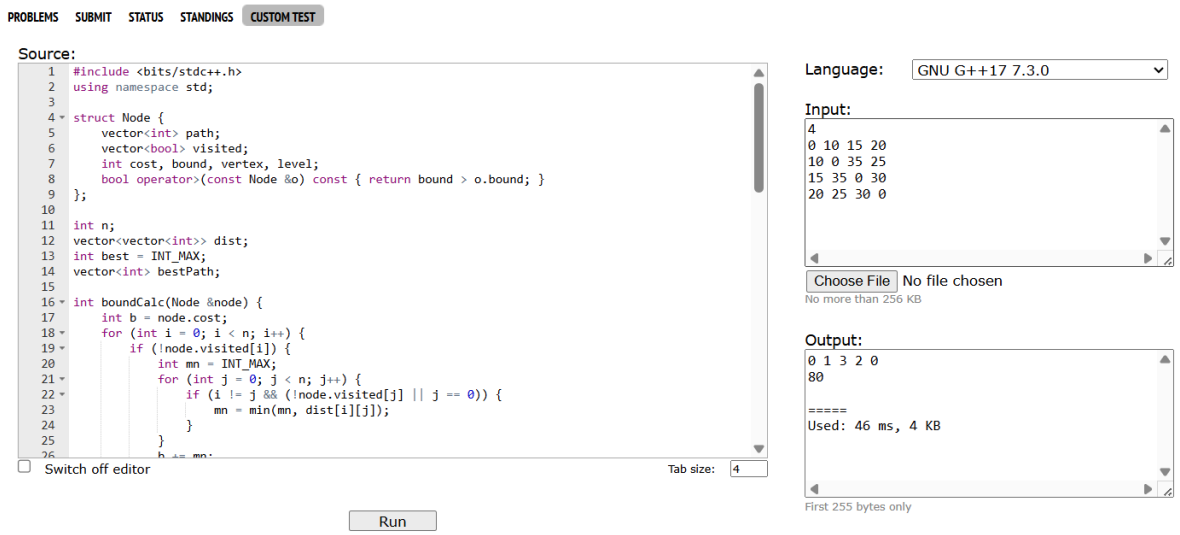
Assignment 4

Problem Statement : Write a program to solve the travelling salesman problem and to print the path and the cost using LC Branch and Bound.

```
#include <bits/stdc++.h>
using namespace std;

struct Node {
    vector<int> path;
    vector<bool> visited;
    int cost, bound, vertex, level;
    bool operator>(const Node &o) const { return bound > o.bound; }
};

int n;
vector<vector<int>> dist;
int best = INT_MAX;
vector<int> bestPath;

int boundCalc(Node &node) {
    int b = node.cost;
    for (int i = 0; i < n; i++) {
        if (!node.visited[i]) {
            int mn = INT_MAX;
            for (int j = 0; j < n; j++) {
                if (i != j && (!node.visited[j] || j == 0)) {
                    mn = min(mn, dist[i][j]);
                }
            }
            b += mn;
        }
    }
    return b;
}

void solveTSP() {
    priority_queue<Node, vector<Node>, greater<Node>> pq;
    Node root;
    root.path.push_back(0);
    root.visited.assign(n, false);
    root.visited[0] = true;
    root.cost = 0;
```

```
    root.vertex = 0;
    root.level = 1;
    root.bound = boundCalc(root);
    pq.push(root);

    while (!pq.empty()) {
        Node cur = pq.top(); pq.pop();
        if (cur.bound >= best) continue;

        if (cur.level == n) {
            int finalCost = cur.cost + dist[cur.vertex][0];
            if (finalCost < best) {
                best = finalCost;
                bestPath = cur.path;
                bestPath.push_back(0);
            }
            continue;
        }

        for (int j = 0; j < n; j++) {
            if (!cur.visited[j]) {
                Node child = cur;
                child.path.push_back(j);
                child.cost += dist[cur.vertex][j];
                child.vertex = j;
                child.level++;
                child.visited[j] = true;
                child.bound = boundCalc(child);
                if (child.bound < best) pq.push(child);
            }
        }
    }
}

int main() {
    cin >> n;
    dist.assign(n, vector<int>(n));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            cin >> dist[i][j];
    solveTSP();
    for (int x : bestPath) cout << x << " ";
    cout << "\n" << best << "\n";
}
```

Test Case 1:



Test Case 2: