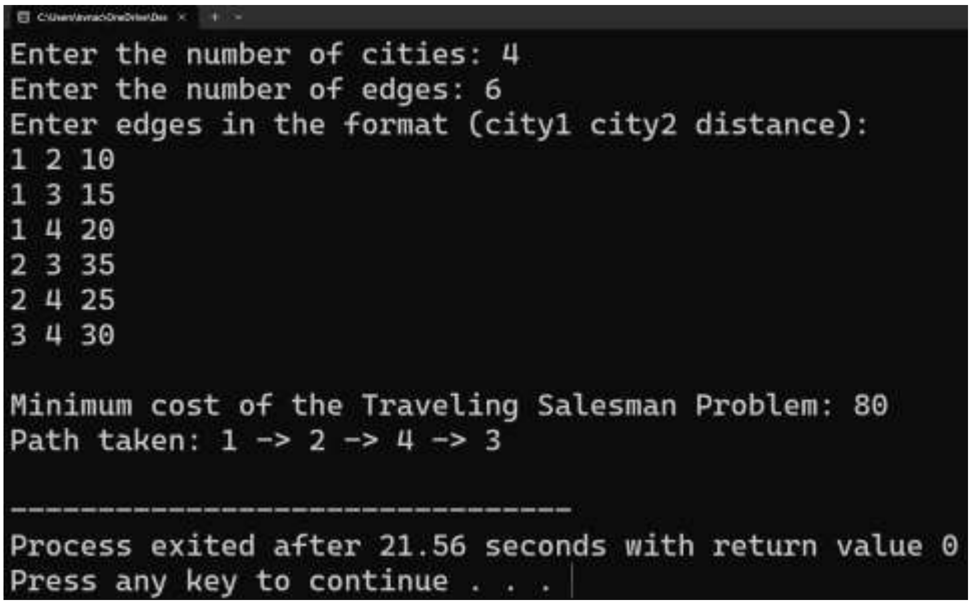
Assignment 10

```
#include <iostream>
#include <climits>
using namespace std;

int numCities;           // Number of cities
int distanceMatrix[20][20]; // Distance matrix
bool visited[20];        // Visited cities tracker
int minCost = INT_MAX;   // Minimum cost found
int bestPath[20];        // Best path found
int currentPath[20];     // Current path being explored

// Function to calculate the cost of the current path
int calculateCost(int path[], int numVisited) {
   int cost = 0;
   for (int i = 0; i < numVisited - 1; i++) {
      cost += distanceMatrix[path[i]][path[i + 1]];
   }
   cost += distanceMatrix[path[numVisited - 1]][path[0]]; // Return to starting city
   return cost;
}

// Branch and Bound function to find the minimum cost path
void branchAndBound(int currentCity, int numVisited) {
   // If all cities have been visited
   if (numVisited == numCities) {
      int cost = calculateCost(currentPath, numVisited);
      if (cost < minCost) {
         minCost = cost; // Update minimum cost
         for (int i = 0; i < numVisited; i++) {
            bestPath[i] = currentPath[i]; // Update best path
            }
        }
        return;
    }

    // Explore other cities
    for (int nextCity = 0; nextCity < numCities; nextCity++) {
        if (!visited[nextCity]) { // If the city has not been visited
            visited[nextCity] = true; // Mark city as visited
            currentPath[numVisited] = nextCity; // Add city to current path

            // Recur to the next city
            branchAndBound(nextCity, numVisited + 1);

            visited[nextCity] = false; // Backtrack
        }
    }
}

int main() {
    // Input number of cities
    cout << "Enter the number of cities: ";
    cin >> numCities;

    // Initialize the distance matrix with infinity
    for (int i = 0; i < numCities; i++) {
        for (int j = 0; j < numCities; j++) {
            distanceMatrix[i][j] = (i == j) ? 0 : INT_MAX; // Distance to itself is 0
        }
    }
    // Input edges and distances
    int edges;
    cout << "Enter the number of edges: ";
    cin >> edges;
    cout << "Enter edges in the format (city1 city2 distance):\n";
    for (int i = 0; i < edges; i++) {
        int city1, city2, distance;
        cin >> city1 >> city2 >> distance;
        distanceMatrix[city1 - 1][city2 - 1] = distance; // Convert to 0-based index
        distanceMatrix[city2 - 1][city1 - 1] = distance; // Undirected graph
    }

    // Start from the first city (0)
    visited[0] = true; // Mark the starting city as visited
    currentPath[0] = 0; // Start from city 0
    branchAndBound(0, 1); // Call the branch and bound function

    // Output the results
    cout << "\nMinimum cost of the Traveling Salesman Problem: " << minCost << endl;
    cout << "Path taken: ";
    for (int i = 0; i < numCities; i++) {
        cout << bestPath[i] + 1; // Output the path (1-based indexing)
        if (i < numCities - 1) cout << " -> ";
    }
    cout << endl;

    return 0;
}
```

Output:



```
Enter the number of cities: 4
Enter the number of edges: 6
Enter edges in the format (city1 city2 distance):
1 2 10
1 3 15
1 4 20
2 3 35
2 4 25
3 4 30

Minimum cost of the Traveling Salesman Problem: 80
Path taken: 1 -> 2 -> 4 -> 3


--------------------------------
Process exited after 21.56 seconds with return value 0
Press any key to continue . . .
```