Name: Sapate Vaibhav Ramdas

Roll No: 307B055

Division: 2

Batch: C

Assignment No: 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.

Program: (With proper comments):

```
#include <iostream>
#include <climits>
using namespace std;
#define N 5 // Number of cities
// Matrix representation of the graph
int M[N][N] = {
    {0, 20, 30, 10, 11},
    {15, 0, 16, 4, 2},
    {3, 5, 0, 2, 4},
    {14, 6, 18, 0, 3},
    {16, 4, 7, 16, 0}};
int cost = INT_MAX; // Initialize cost as maximum value
int best_path[N];   // Array to store the best path
// Function to solve the TSP problem using Branch and Bound
void tsp_branch_and_bound(int path[N], bool visited[N], int bound, int level)
{
```

```
// If all cities have been visited
if (level == N)
{
   // Calculate the cost of the current path
   int current_cost = bound + M[path[N - 1]][path[0]];
   // If the current cost is less than the minimum cost found so far
   if (current_cost < cost)
   {
      // Update the minimum cost and the best path
      cost = current_cost;
      copy(path, path + N, best_path);
   }
   return;
}
// For each city
for (int i = 0; i < N; ++i)
{
   // If the city has not been visited yet
   if (!visited[i])
   {
      // Calculate the bound for the next level
      int new_bound = bound + M[path[level - 1]][i];
      // If the new bound is less than the minimum cost found so far
      if (new_bound < cost)
      {
         // Mark the city as visited and go to the next level
         path[level] = i;
```

```
                visited[i] = true;
                tsp_branch_and_bound(path, visited, new_bound, level + 1);
                // Backtrack: mark the city as not visited for future iterations
                visited[i] = false;
            }
        }
    }
}
int main(){
    int path[N];                // Array to store the current path
    bool visited[N] = {false}; // Boolean array to keep track of visited cities
    path[0] = 0;       // Start from city 0
    visited[0] = true; // Mark city 0 as visited
    tsp_branch_and_bound(path, visited, 0, 1); // Call the TSP function
    cout << "Min Cost: " << cost << endl; // Print the minimum cost
    cout << "Best Path: "; // Print the best path
    for (int i = 0; i < N; ++i)
    {
        cout << best_path[i] << " --> ";
    }
    cout << best_path[0] << endl; // Print the starting city to complete the cycle
    return 0;
}
```

Output:

```
Min Cost: 28
Best Path: 0 --> 3 --> 1 --> 4 --> 2 --> 0
```