“**Experiment 2.2”**

**Aim:**

To demonstrate the concept of Graph.

**Objective:**

• The objective is to build problem solving capability and to learn the basic concepts of data structures.

• The implementation of Find the difference problem brushes up the concept of graph.

• The implementation of Predict the winner problem brushes up the concept of graph.

**Problem 1: “Find the difference”**

<https://leetcode.com/problems/find-the-difference/description/>

You are given two strings s and t.

String t is generated by random shuffling string s and then add one more letter at a random position.

Return the letter that was added to t.

**Code:**

class Solution {

public:

    char findTheDifference(string s, string t) {

        char c= 0;

        for(char i: s) c ^= i;

        for(char i: t) c ^= i;

        return c;

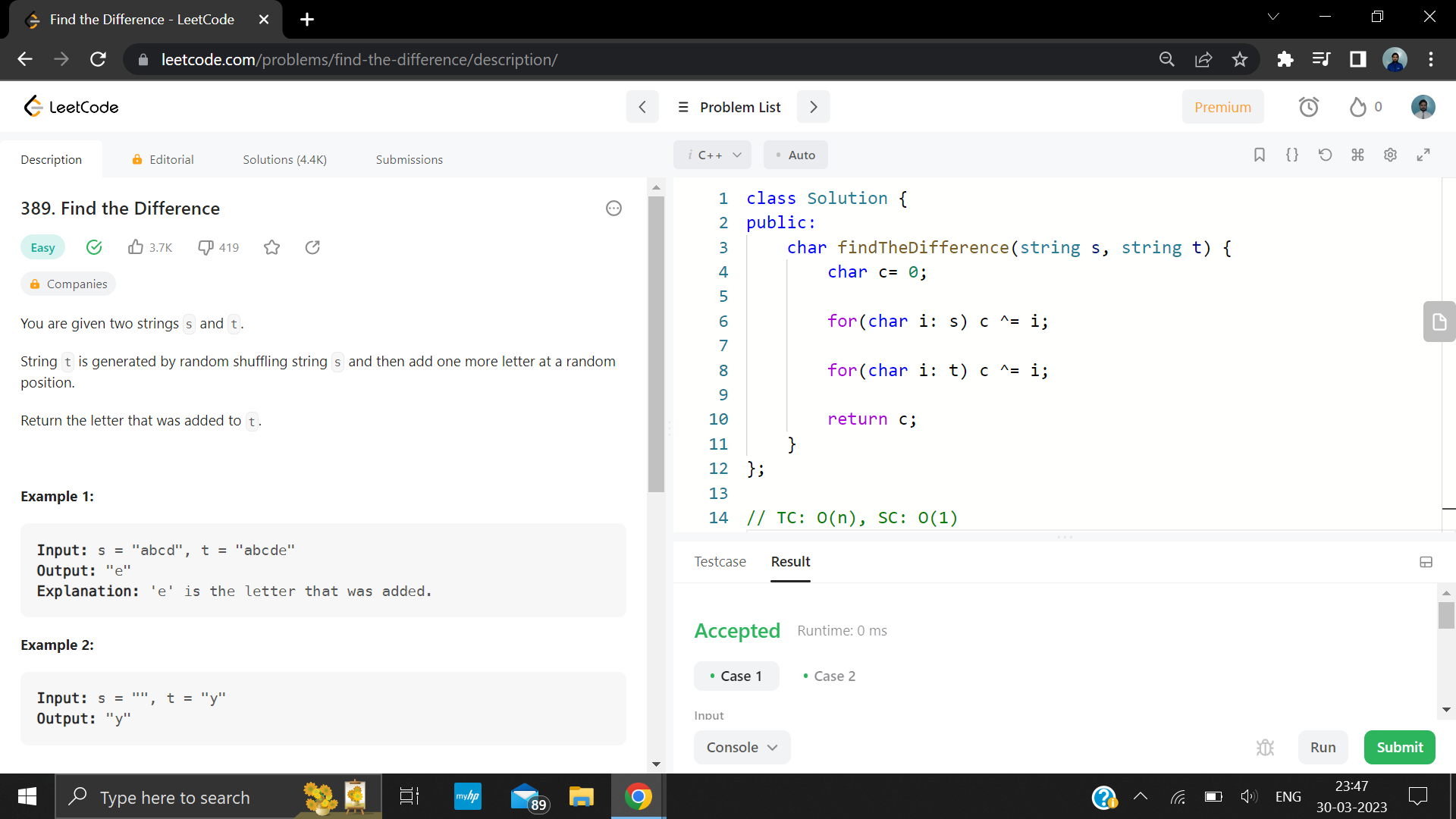
    }

};

// Time Complexity: O(n)

// Space Complexity: O(1)

**Output:**



**Problem 2: “Predict the winner”**

<https://leetcode.com/problems/predict-the-winner/description/>

You are given an integer array nums. Two players are playing a game with this array: player 1 and player 2.

Player 1 and player 2 take turns, with player 1 starting first. Both players start the game with a score of 0. At each turn, the player takes one of the numbers from either end of the array (i.e., nums[0] or nums[nums.length - 1]) which reduces the size of the array by 1. The player adds the chosen number to their score. The game ends when there are no more elements in the array.

Return true if Player 1 can win the game. If the scores of both players are equal, then player 1 is still the winner, and you should also return true. You may assume that both players are playing optimally.

**Code:**

class Solution {

public:

    bool PredictTheWinner(vector<int>& nums) {

        int n = nums.size();

        vector<vector<int>> dp(n, vector<int>(n)); // use to keep the score gap between player1 and player2

        for (int i = 0; i < n; i++) dp[i][i] = nums[i];

        for (int i = 1; i < n; i++) {

            for (int j = 0; j+i < n; j++) {

                dp[j][j+i] = max(nums[j+i]-dp[j][j+i-1], nums[j]-dp[j+1][j+i]);

            }

        }

        return dp[0][n-1] >= 0; // player1 get more score points than player2

    }

};

**Output:**

