“**Experiment 2.3”**

**Aim:**

To demonstrate the concept of Divide and Conquer.

**Objective:**

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

• The implementation of Count and Say problem brushes up the concept of divide and conquer.

• The implementation of Water and Jug problem brushes up the concept of divide and conquer.

**Problem 1: “Count and Say”**

<https://leetcode.com/problems/count-and-say/description/>

The count and say sequence is a sequence of digit strings defined by the recursive formula:

* countAndSay(1) = "1"
* countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string.

To determine how you "say" a digit string, split it into the minimal number of substrings such that each substring contains exactly one unique digit. Then for each substring, say the number of digits, then say the digit. Finally, concatenate every said digit.

**Code:**

class Solution {

public:

    string countAndSay(int n) {

        // base cases

        if(n==1) return "1";

        if(n==2) return "11";

        string s= "11";

        for(int i=3;i<=n;i++){

            string t= "";

            s= s+'$';

            int cnt= 1;

            for(int j=1;j<s.length();j++){

                if(s[j]!=s[j-1]){

                    t= t + to\_string(cnt);

                    t= t + s[j-1];

                    cnt= 1;

                }

                else cnt++;

            }

            s= t;

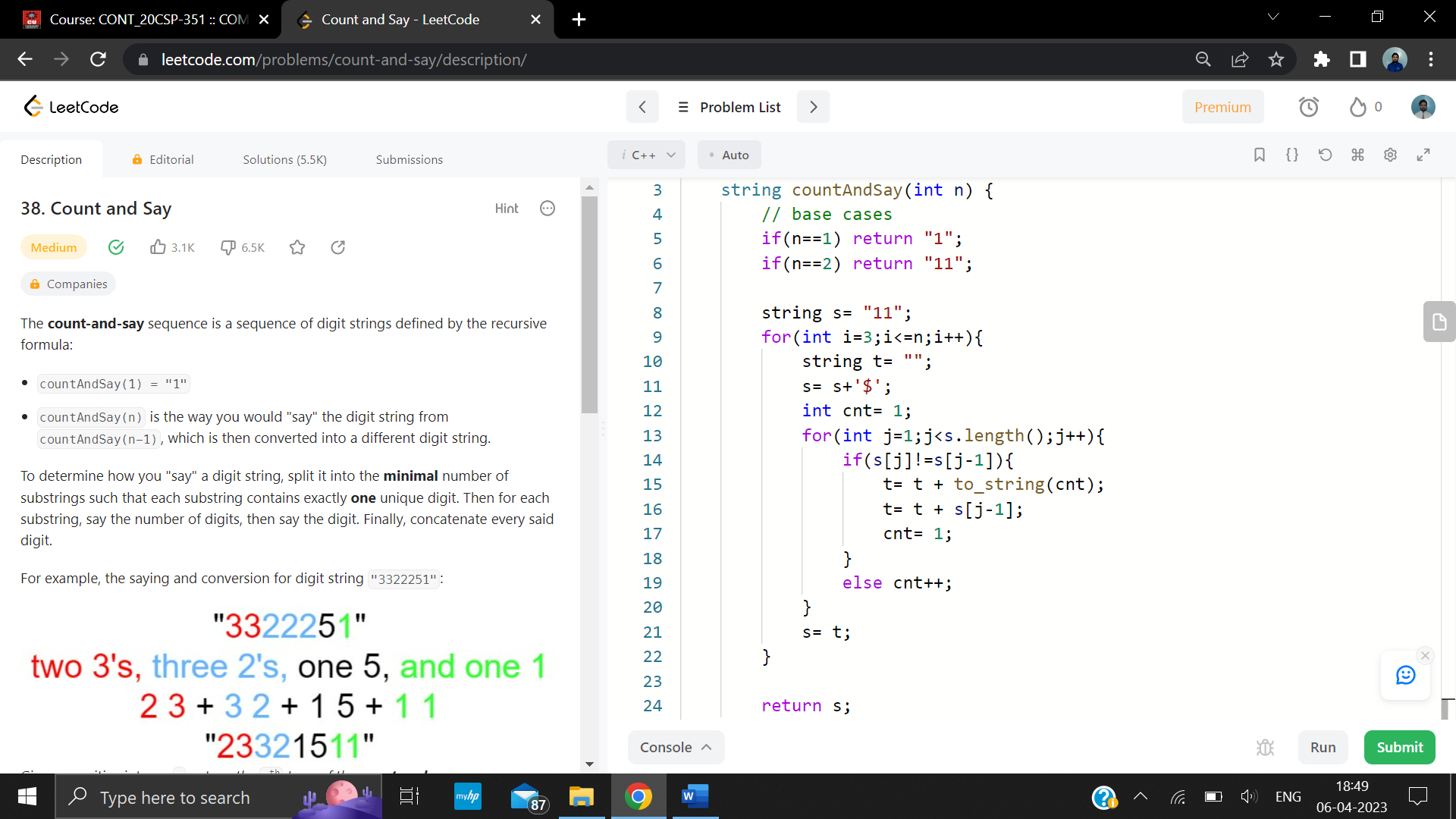
        }

        return s;

    }

};

**Output:**



**Problem 2: “Water and Jug”**

<https://leetcode.com/problems/water-and-jug-problem/description/>

You are given two jugs with capacities jug1Capacity and jug2Capacity liters. There is an infinite amount of water supply available. Determine whether it is possible to measure exactly targetCapacity liters using these two jugs.

If targetCapacity liters of water are measurable, you must have targetCapacity liters of water contained within one or both buckets by the end.

Operations allowed:

* Fill any of the jugs with water.
* Empty any of the jugs.
* Pour water from one jug into another till the other jug is completely full, or the first jug itself is empty.

**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:**

