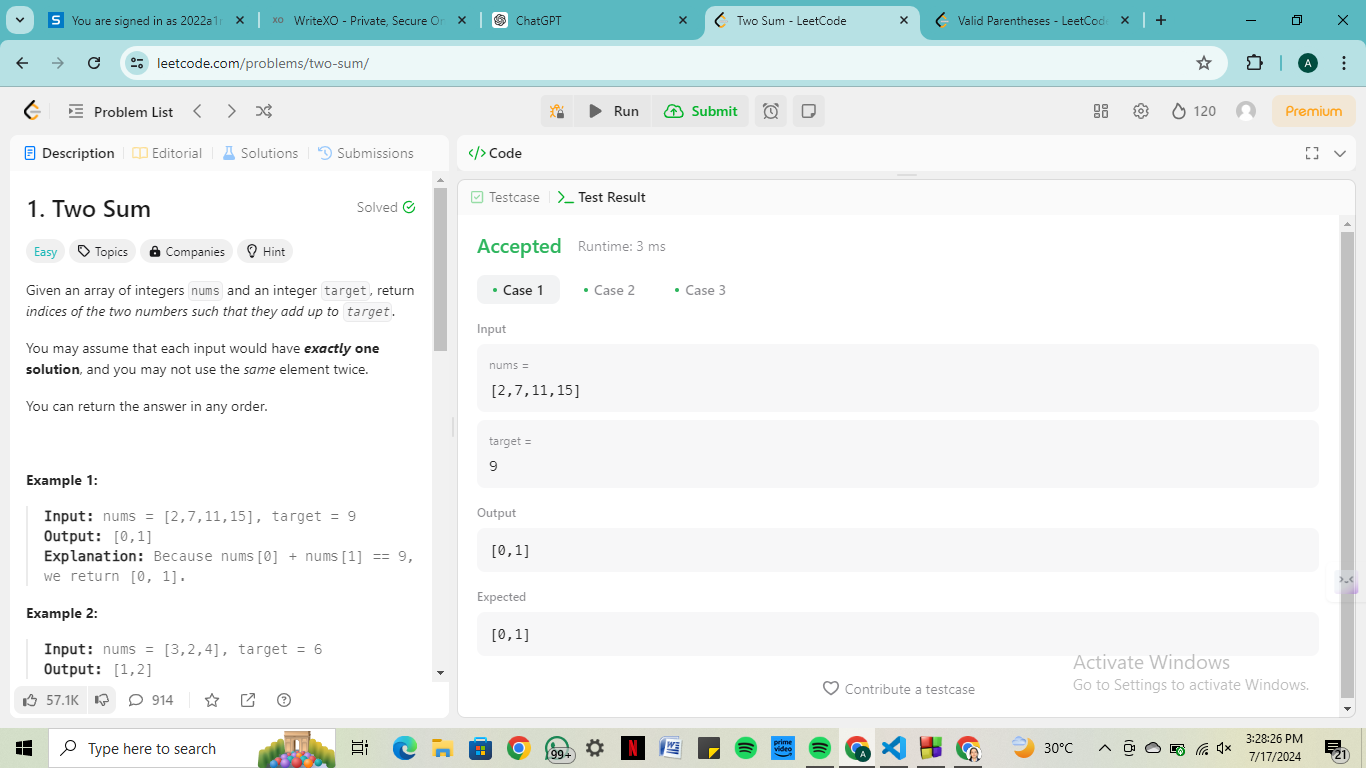
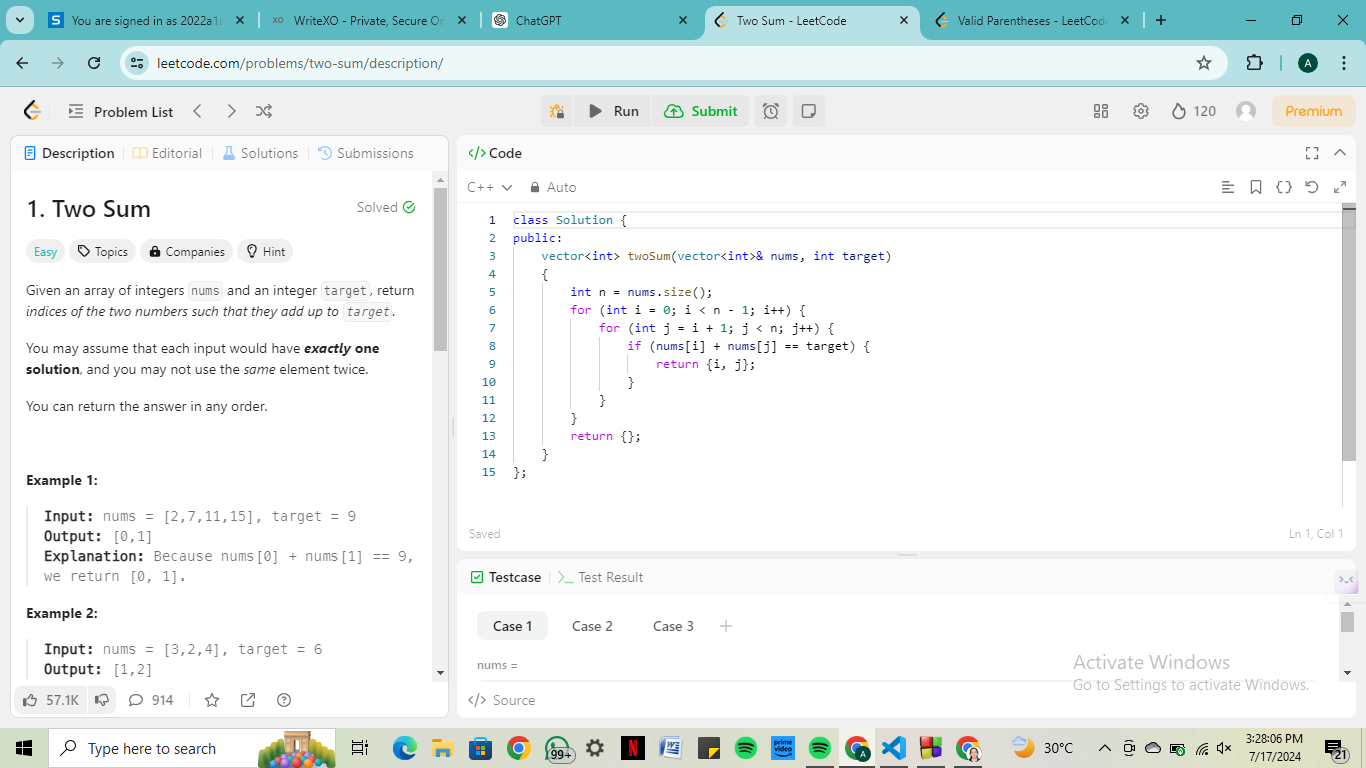
**TWO SUM**

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to *target*.

You may assume that each input would have **exactly one solution**, and you may not use the same element twice.

You can return the answer in any order.

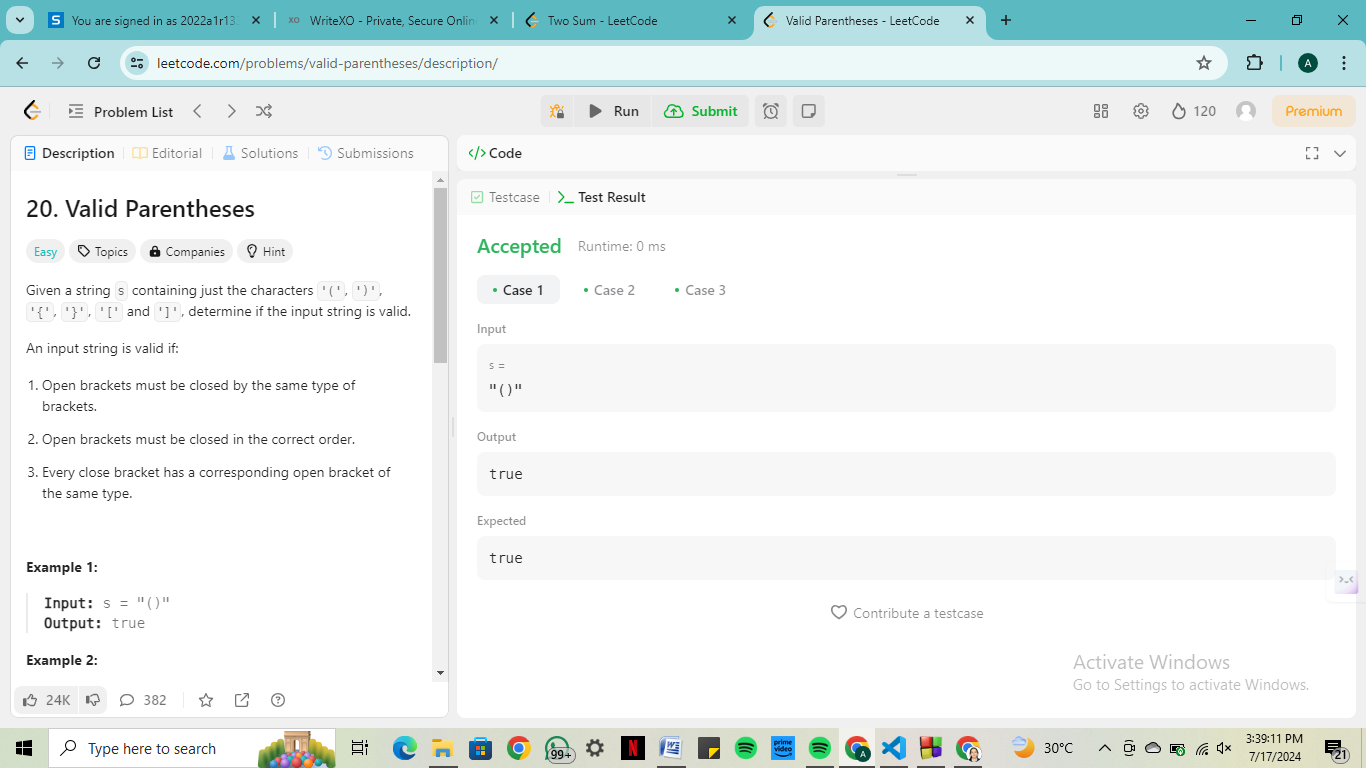
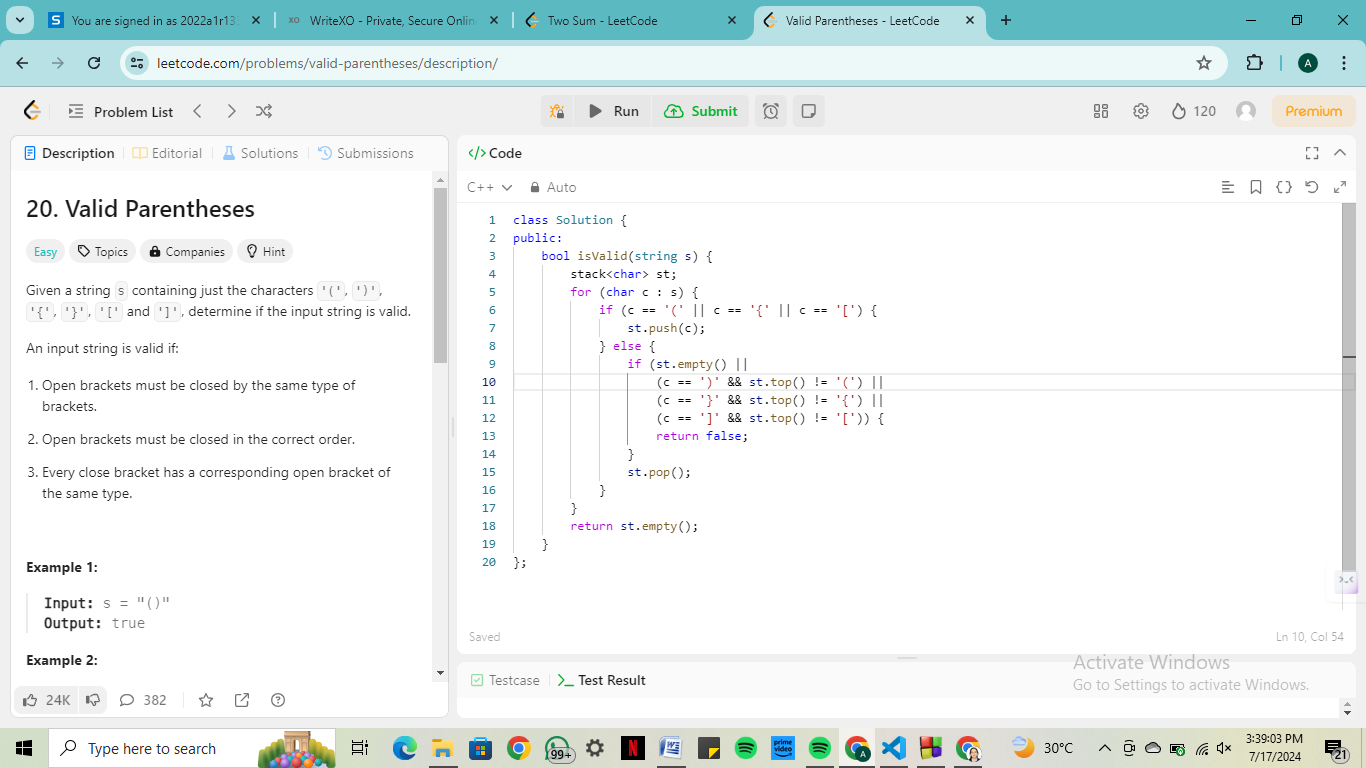


**VALID PARENTHESIS**

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.



**Two Sum**

class Solution {

public:

    vector<int> twoSum(vector<int>& nums, int target)

    {

        int n = nums.size();

        for (int i = 0; i < n - 1; i++) {

            for (int j = i + 1; j < n; j++) {

                if (nums[i] + nums[j] == target) {

                    return {i, j};

                }

            }

        }

        return {};

    }

};

**Valid Parenthesis**

class Solution {

public:

    bool isValid(string s) {

        stack<char> st;

        for (char c : s) {

            if (c == '(' || c == '{' || c == '[') {

                st.push(c);

            } else {

                if (st.empty() ||

                    (c == ')' && st.top() != '(') ||

                    (c == '}' && st.top() != '{') ||

                    (c == ']' && st.top() != '[')) {

                    return false;

                }

                st.pop();

            }

        }

        return st.empty();

    }

};