“**Experiment 2.1”**

**Aim:**

To demonstrate the concept of Trees.

**Objective:**

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

• The implementation of Balanced Binary Tree problem brushes up the concept of trees.

• The implementation of Path Sum problem brushes up the concept of trees.

**Problem 1: “Balanced Binary Tree”**

<https://leetcode.com/problems/balanced-binary-tree/>

Given a binary tree, determine if it is height-balanced.

**Code:**

class Solution {

public:

    bool isBalanced(TreeNode\* root) {

        return dfsHeight(root)!=-1;

    }

    int dfsHeight(TreeNode\* root){      // O(n)

        // base case

        if(root==NULL) return 0;

        int leftHeight= dfsHeight(root->left);

        int rightHeight= dfsHeight(root->right);

        if(abs(leftHeight-rightHeight) > 1) return -1;

        if(leftHeight==-1 || rightHeight==-1) return -1;

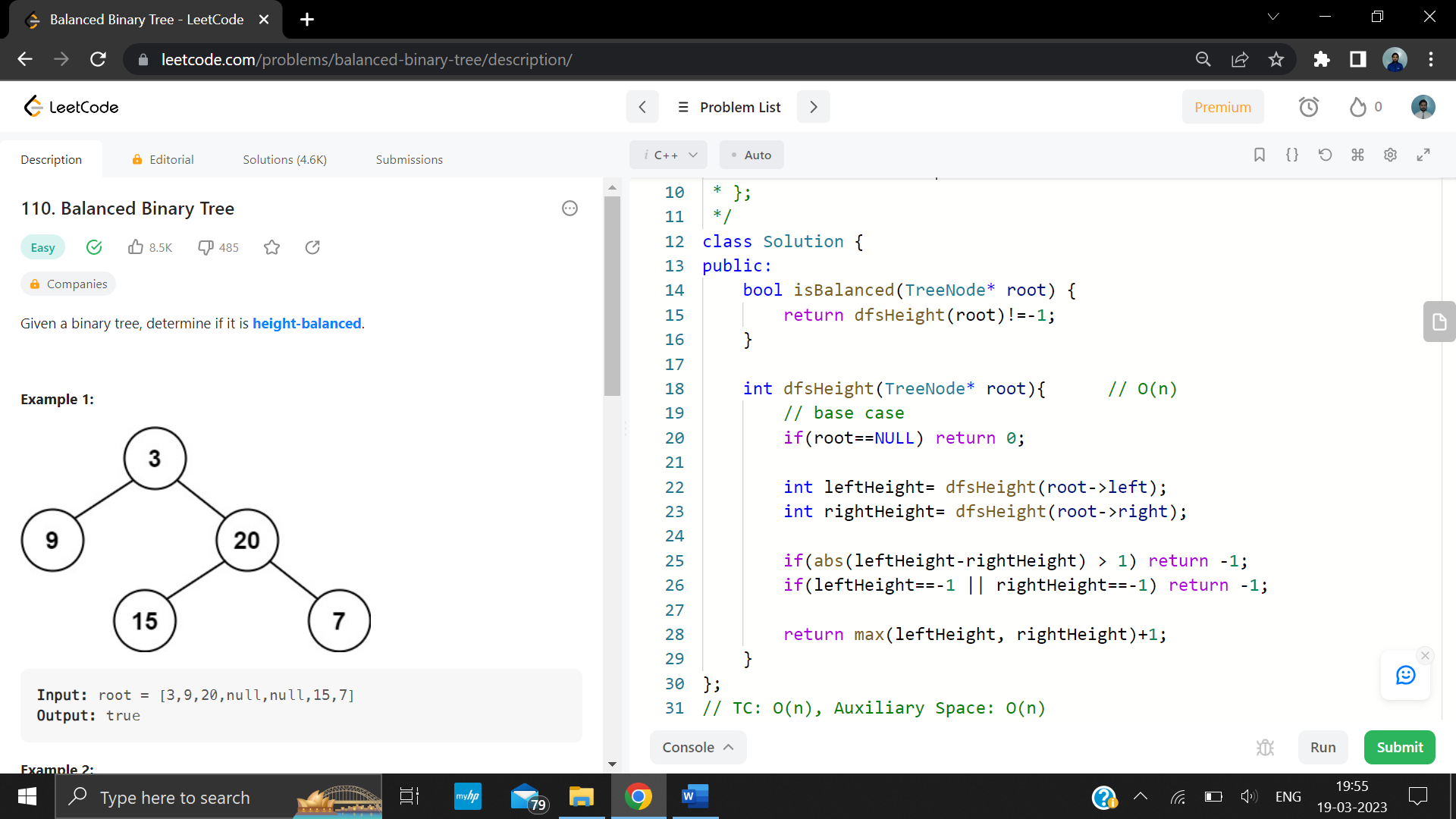
        return max(leftHeight, rightHeight)+1;

    }

};

// TC: O(n), Auxiliary Space: O(n)

**Output:**



**Problem 2: “Path Sum”**

<https://leetcode.com/problems/path-sum/description/>

Given the root of a binary tree and an integer targetSum, return true if the tree has a root-to-leaf path such that adding up all the values along the path equals targetSum.

A leaf is a node with no children.

**Code:**

class Solution {

public:

    bool hasPathSum(TreeNode\* root, int targetSum) {        // TC: O(n)

        // base case

        if(root==NULL) return false;

        if(root->left==NULL && root->right==NULL) return (targetSum - root->val) == 0;

        return hasPathSum(root->left, targetSum - root->val) || hasPathSum(root->right, targetSum - root->val);

    }

};

// TC: O(n), Auxiliary Space: O(n)

**Output:**

