SUMMER TRAINING COURSE-MINOR PROJECT REPORT

LOVELY PROFESSIONAL UNIVERSITY

COURSE BY-BOARD INFINITY

NAME – SHASHANK SHEKHAR

COURSE- BTECH CSE

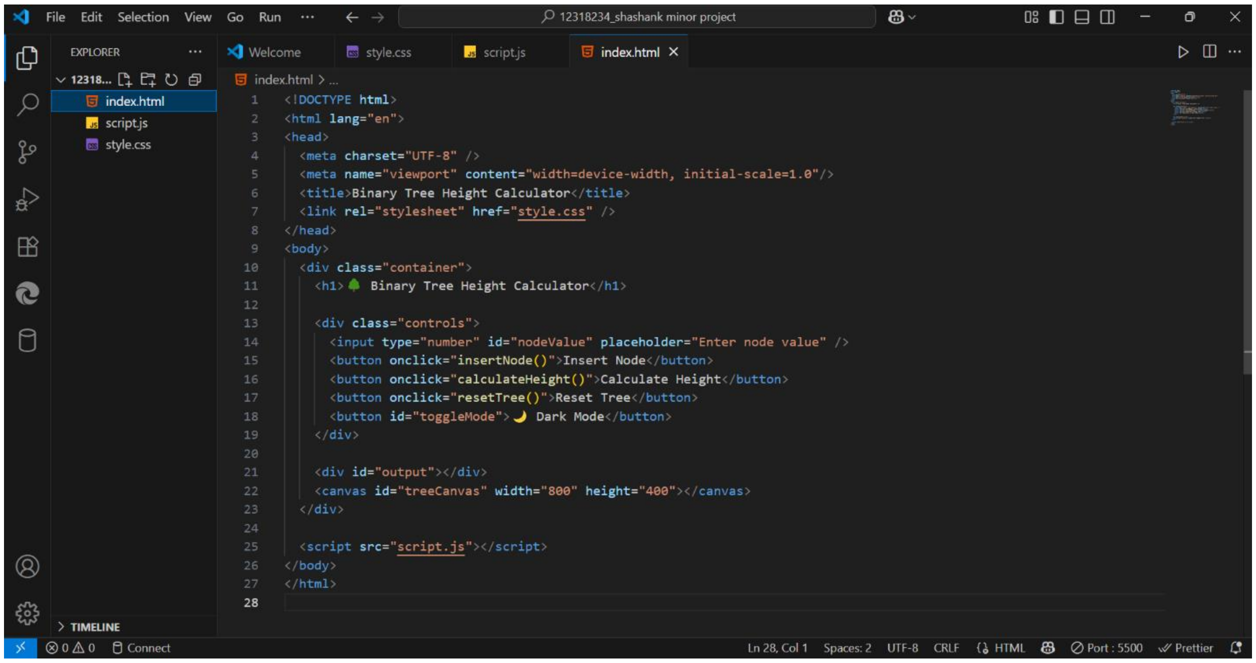
REGISTRATION NUMBER-12318234

PROJECT TYPE-MINOR PROJECT

TOPIC-HEIGHT OF BINARY TREE

SUBJECT-DSA FOR INTERVIEW

HTML CODE AND EXPLANATION :



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Binary Tree Height Calculator</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <div class="container">
    <h1>🌳 Binary Tree Height Calculator</h1>

    <div class="controls">
      <input type="number" id="nodeValue" placeholder="Enter node value" />
      <button onclick="insertNode()">Insert Node</button>
      <button onclick="calculateHeight()">Calculate Height</button>
      <button onclick="resetTree()">Reset Tree</button>
      <button id="toggleMode">🌙 Dark Mode</button>
    </div>

    <div id="output"></div>
    <canvas id="treeCanvas" width="800" height="400"></canvas>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

EXPLANATION OF CODE:

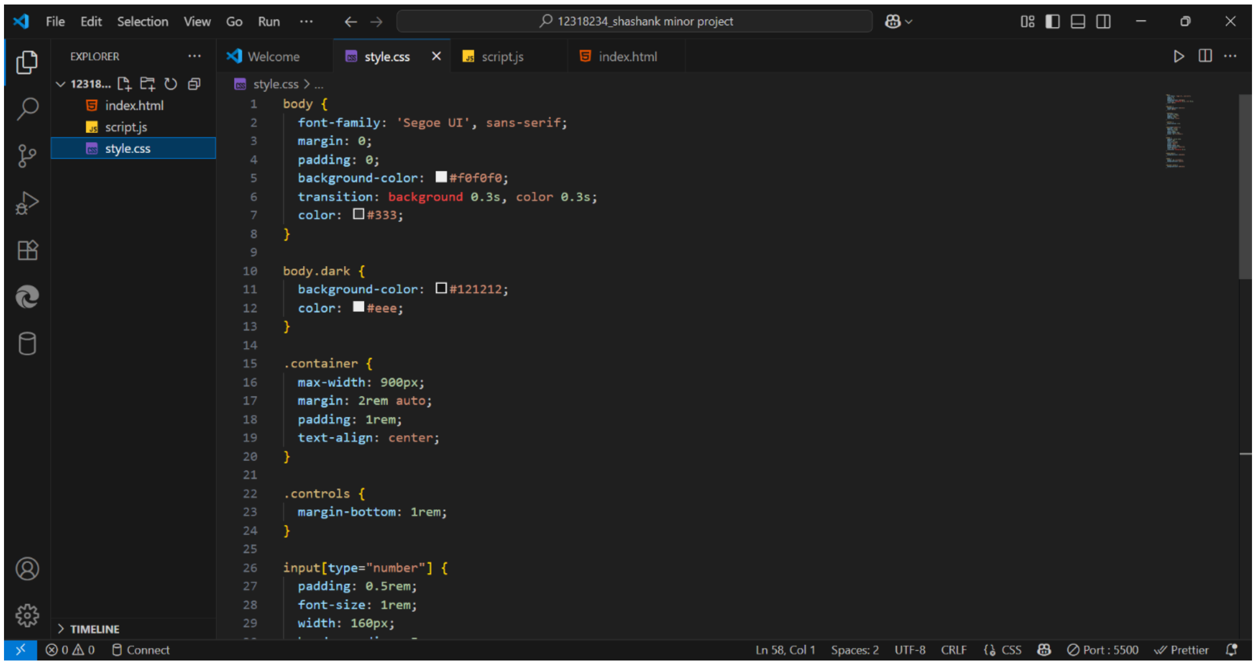
1. index.html – Structure & Layout of Our Web App

This is the main HTML file where we define the layout of our Binary Tree Height Calculator.

Key Components:

- **Title & Heading**: Displays the name of our app.
- **Input & Buttons**:
  - Input box to enter node values
  - Buttons to insert nodes, calculate tree height, reset the tree, and toggle dark mode
- **Output Area**: Shows the calculated height of the tree
- **Canvas**: Visualizes our binary tree structure
- **Script Link**: Connects our HTML to the JavaScript logic file

CSS CODE AND EXPLANATION:

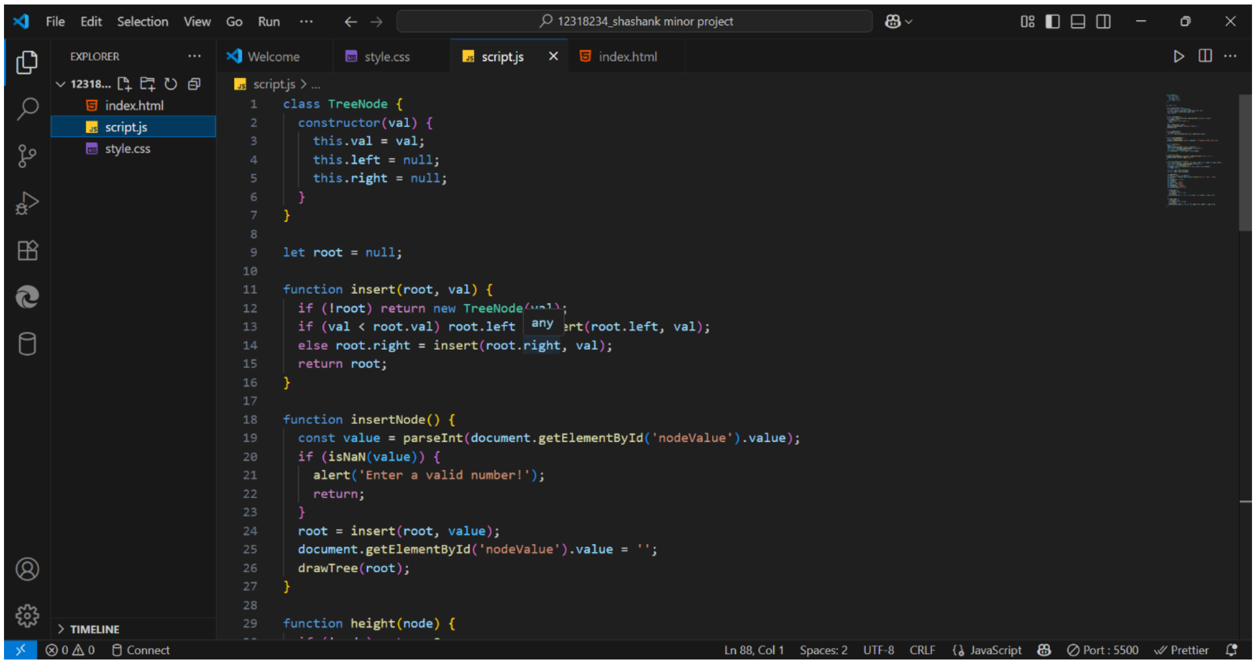


```css
body {
  font-family: 'Segoe UI', sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f0f0f0;
  transition: background 0.3s, color 0.3s;
  color: #333;
}

body.dark {
  background-color: #121212;
  color: #eee;
}

.container {
  max-width: 900px;
  margin: 2rem auto;
  padding: 1rem;
  text-align: center;
}

.controls {
  margin-bottom: 1rem;
}

input[type="number"] {
  padding: 0.5rem;
  font-size: 1rem;
  width: 160px;
```

EXPLANATION: **2. style.css – Styling & Theme of Our website**
This file makes our website look visually clean and modern.
Key Styling Features:

- **Light/Dark Mode Support**: Switches background and text colors smoothly
- **Buttons & Inputs**: Styled for easy interaction and good appearance
- **Responsive Design**: Ensures our layout looks neat on all devices
- **Canvas Styling**: Adds border and background that change with theme

JAVASCRIPT CODE AND EXPLANATION:


File Edit Selection View Go Run ···
12318234_shashank minor project
EXPLORER
12318...
index.html
script.js
style.css
Welcome
style.css
script.js
index.html
script.js > ...
TIMELINE
0 0 Connect
Ln 88, Col 1 Spaces: 2 UTF-8 CRLF JavaScript Port : 5500 Prettier

```
class TreeNode {
  constructor(val) {
    this.val = val;
    this.left = null;
    this.right = null;
  }
}

let root = null;

function insert(root, val) {
  if (!root) return new TreeNode(val);
  if (val < root.val) root.left  any  ert(root.left, val);
  else root.right = insert(root.right, val);
  return root;
}

function insertNode() {
  const value = parseInt(document.getElementById('nodeValue').value);
  if (isNaN(value)) {
    alert('Enter a valid number!');
    return;
  }
  root = insert(root, value);
  document.getElementById('nodeValue').value = '';
  drawTree(root);
}

function height(node) {
```

EXPLANATION:

3. script.js – Tree Logic & Interactivity in Our App

This file handles the main logic and interactivity for our binary tree.

Main Functions:

- **TreeNode Class**: Defines the structure of each node in our tree
- **insert()**: Adds values to our binary search tree
- **height()**: Calculates the maximum depth (height) of our tree

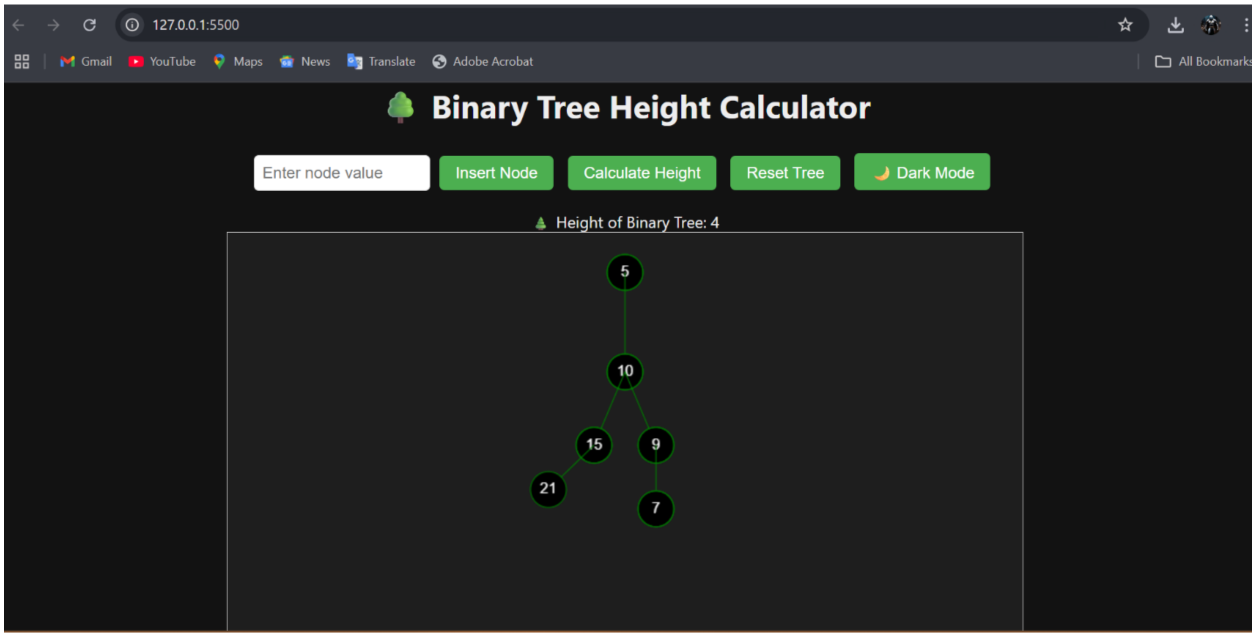
User Interaction:

- **insertNode()**: Takes user input and inserts a node into our tree
- **calculateHeight()**: Computes and displays the tree's height
- **resetTree()**: Clears our tree, canvas, and output

Additional Features:

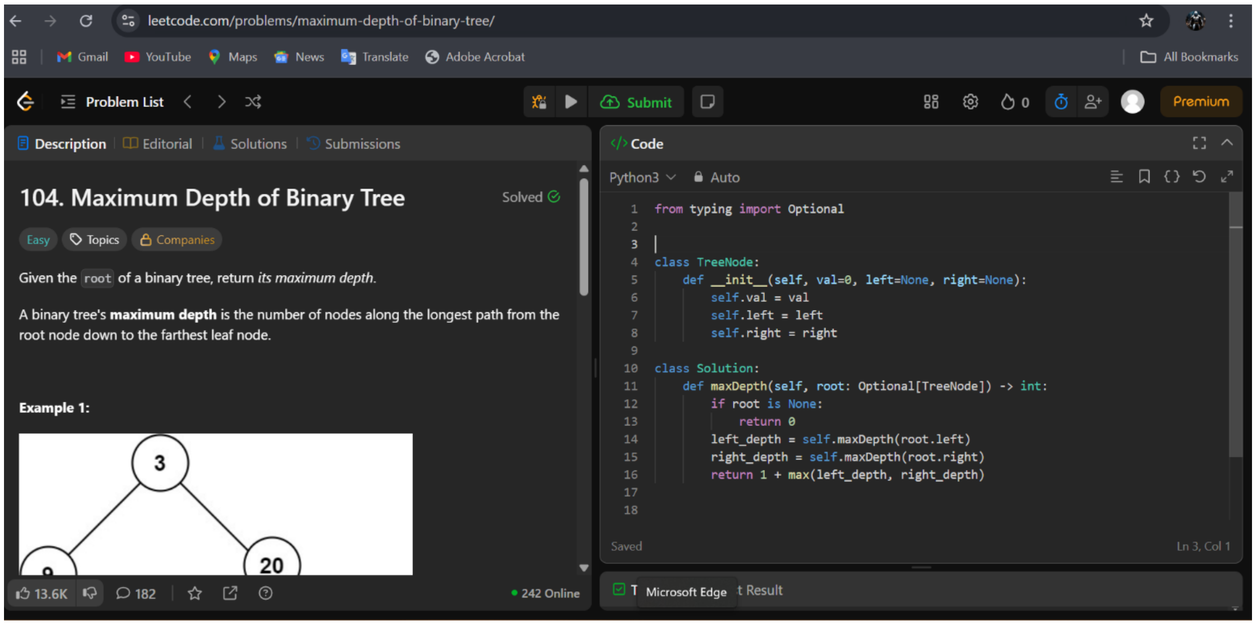
- **Dark Mode Toggle**: Lets us switch between light and dark themes
- **Tree Visualization**: Draws our binary tree using the HTML Canvas element

WEBSITE SCREENSHOT:



GITHUB REPOSITORY LINK: https://github.com/sshashank13/height-of-binary-tree-calculator

LEETCODE AND GEEK FOR GEEKS QUESTION SOLVED FOR BETTER UNDERSTANDING OF CONCEPT HEIGHT AND DEPTH OF BINARY TREE:


leetcode.com/problems/maximum-depth-of-binary-tree/

104. Maximum Depth of Binary Tree

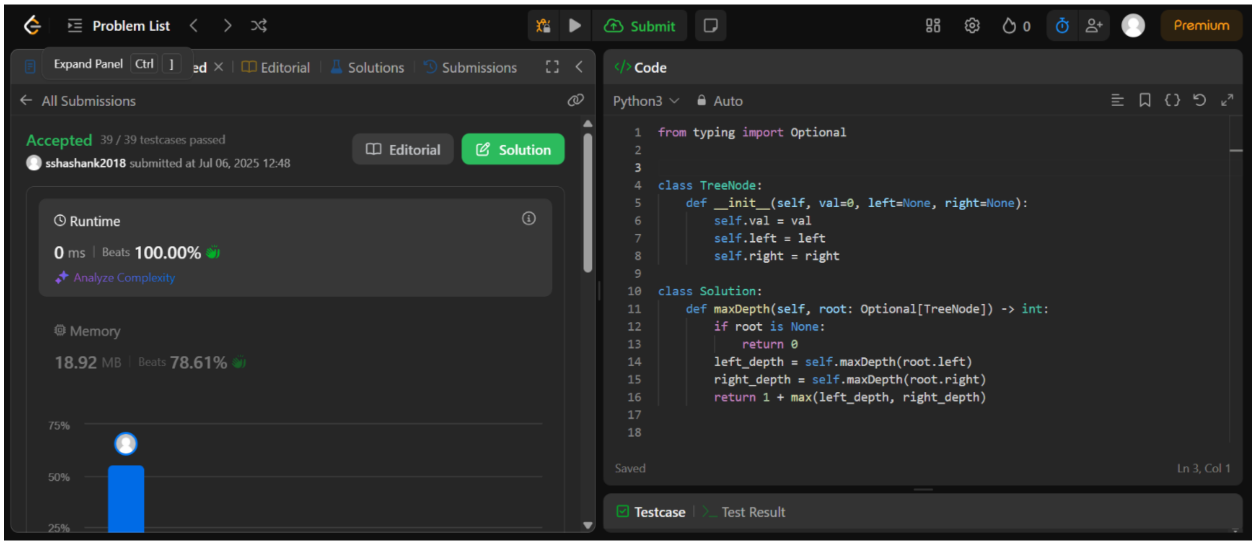
Solved

Given the `root` of a binary tree, return *its maximum depth*.

A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.

Example 1:

```python
from typing import Optional

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def maxDepth(self, root: Optional[TreeNode]) -> int:
        if root is None:
            return 0
        left_depth = self.maxDepth(root.left)
        right_depth = self.maxDepth(root.right)
        return 1 + max(left_depth, right_depth)
```


Accepted 39 / 39 testcases passed

sshashank2018 submitted at Jul 06, 2025 12:48

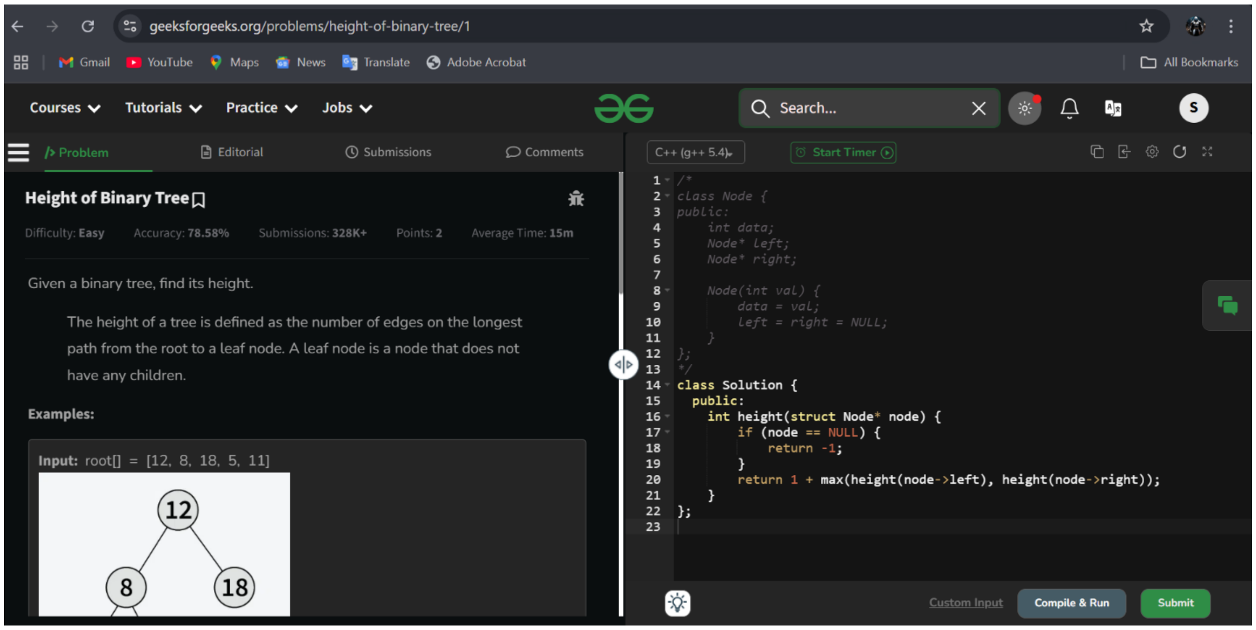
Runtime

0 ms | Beats 100.00%

Memory

18.92 MB | Beats 78.61%

```python
from typing import Optional

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def maxDepth(self, root: Optional[TreeNode]) -> int:
        if root is None:
            return 0
        left_depth = self.maxDepth(root.left)
        right_depth = self.maxDepth(root.right)
        return 1 + max(left_depth, right_depth)
```



geeksforgeeks.org/problems/height-of-binary-tree/1

Height of Binary Tree

Difficulty: Easy Accuracy: 78.58% Submissions: 328K+ Points: 2 Average Time: 15m

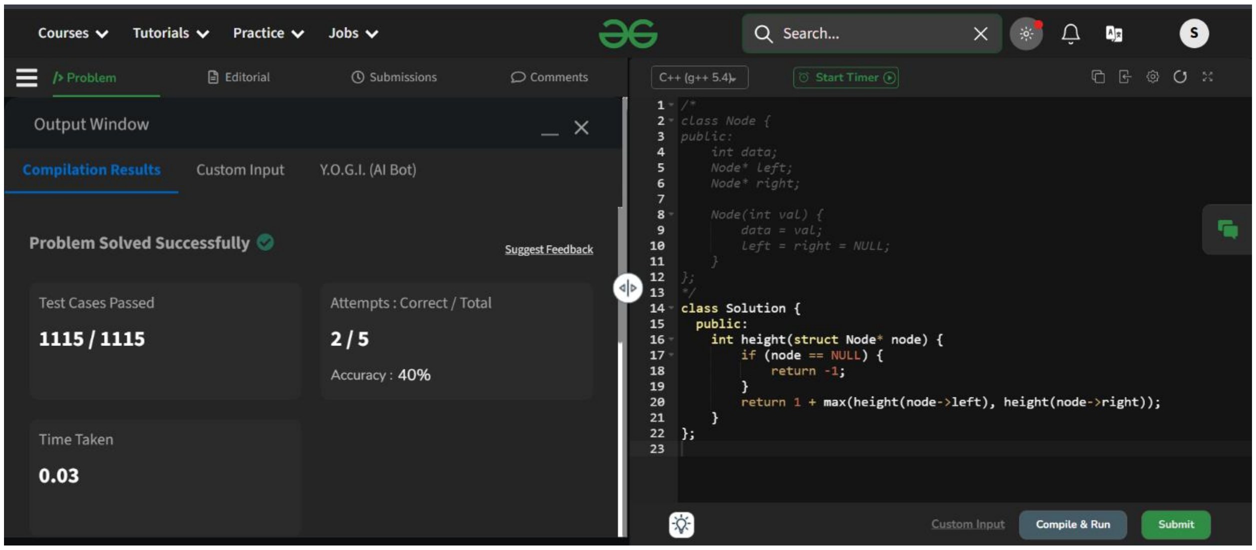
Given a binary tree, find its height.

The height of a tree is defined as the number of edges on the longest path from the root to a leaf node. A leaf node is a node that does not have any children.

Examples:

Input: root[] = [12, 8, 18, 5, 11]

```cpp
/*
class Node {
public:
    int data;
    Node* left;
    Node* right;

    Node(int val) {
        data = val;
        left = right = NULL;
    }
};
*/
class Solution {
  public:
    int height(struct Node* node) {
        if (node == NULL) {
            return -1;
        }
        return 1 + max(height(node->left), height(node->right));
    }
};
```



Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
2 / 5
Accuracy : 40%

Time Taken
0.03

```cpp
/*
class Node {
public:
    int data;
    Node* left;
    Node* right;

    Node(int val) {
        data = val;
        left = right = NULL;
    }
};
*/
class Solution {
  public:
    int height(struct Node* node) {
        if (node == NULL) {
            return -1;
        }
        return 1 + max(height(node->left), height(node->right));
    }
};
```