Lab 5: Trees

ELEC 278: Fundamentals of Information Structures

Learning Goals

- Define and explain key concepts related to trees, including nodes, edges, root, leaves, height, and depth.
- Apply recursive algorithms to do core operations on trees including traversal and insertion.
- Understand how binary search trees facilitate efficient searching.

Instructions

Download the zipped folder `lab5.zip` from OnQ and unzip it. Next, open the lab5 folder that you extracted in your code editor. You will complete the following exercises by completing the functions in the `main.c` file.

To receive full marks, your code must be able to correctly handle corner cases. Comment your code and be ready to discuss how your code handles corner cases with your TA. Generate a list a cases that you will test your code against with your TA.

Grading (fraction of exercise marks)

0	No code or lack of understanding of the code.
1/2	Functioning code, but fails corner cases, if any, or not readable/not well-explained.
1	Functioning code that handles corner cases

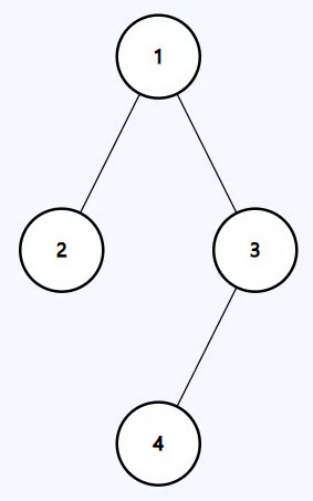
You are given the following `C`-struct for a tree node.

```
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
};
```

Exercise 1. (2 marks) Given the root of a binary tree, write a function `int Height(struct TreeNode* root)` that returns its height.

The height of a binary tree is defined as the number of nodes along the longest path from the root node down to the farthest leaf node.

Assume that the height of a leaf is 0. For example, the height of the tree below is 2.

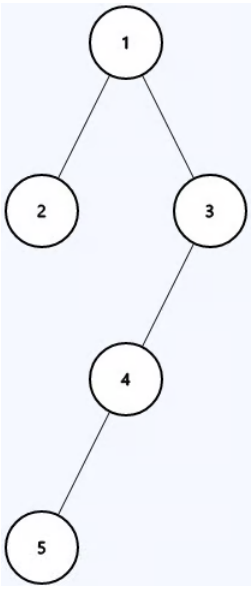


Exercise 2. (2 marks) Given the root of a binary tree, write a function `bool isBalanced(struct TreeNode* root)` that returns true if it is height-balanced and false otherwise.

A height-balanced binary tree is defined as a binary tree in which the left and right subtrees of every node differ in height by no more than 1.

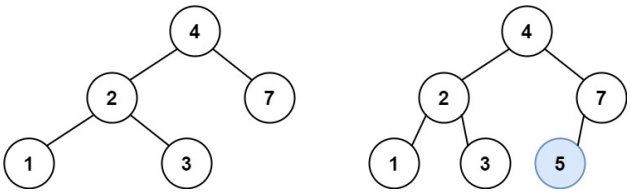
For example, the above figure shows a height-balanced binary tree.

The tree below is not.



Exercise 3. (2 marks) Given the root of a binary tree, return the zigzag level order traversal of its nodes' values. (i.e., from left to right, then right to left for the next level and alternate between).

Example: The zigzag level order traversal of the tree on the left in the figure below is $4, 7, 2, 1, 3$. The zigzag level order traversal of the tree on the right in the figure below is $4, 7, 2, 1, 3, 5$



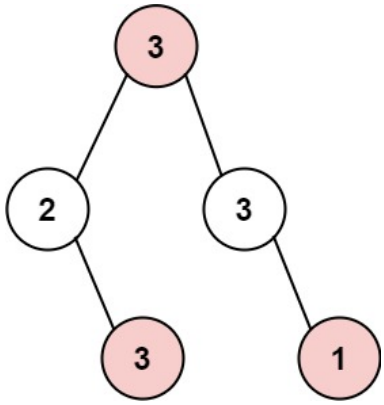
Exercise 4. (4 marks)

The rooms in an ancient Egyptian pyramid are arranged in a binary tree layout. The nodes of the tree represent rooms, and the edges represent corridors between the rooms. You access the pyramid through a root room.

A treasure map that has led you to the pyramid has an instruction to avoid looting two directly linked rooms to avoid awakening the curse of the Pharaoh.

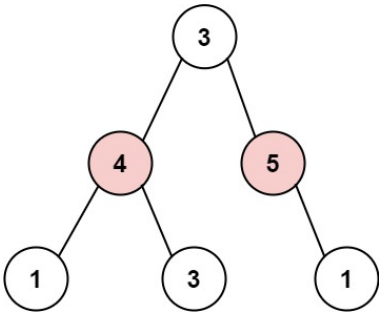
Given the root of the binary tree, the node values represent the number of items in the room. Return the maximum amount of items you can loot from the pyramid.

Example 1:



Maximum amount of items that can be looted = 3 + 3 + 1 = 7.

Example 2:



Maximum amount of items that can be looted = 4 + 5 = 9.

Stolen historical artifacts are irreplaceable pieces of their origin's history. When they are looted, we lose pieces that can help us narrate our human history. The artifacts that have been stolen from

Egypt, India, Nigeria, and other places around the world are countless. In 2021, Egypt saw the return of 5,300 looted artifacts from across the globe (AP reports).