Tree [CO2]

Instructions for students:

- Complete the following methods on Tree.
- You may use any language to complete the tasks.
- All your methods must be written in one single .java or .py or .pynb file.
  DO NOT CREATE separate files for each task.
- If you are using JAVA, you must include the main method as well which
  should test your other methods and print the outputs according to the tasks.
- If you are using PYTHON, then follow the coding templates shared in this

folder.

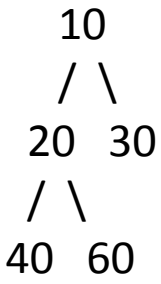
NOTE:

- **YOU CANNOT USE ANY BUILT-IN FUNCTION EXCEPT** len **IN PYTHON. [negative indexing, append is prohibited]**
- **YOU HAVE TO MENTION SIZE OF ARRAY WHILE INITIALIZATION**
- **YOUR CODE SHOULD WORK FOR ALL RELEVANT SAMPLE INPUTS**
- **DO NOT USE LIST, QUEUE**

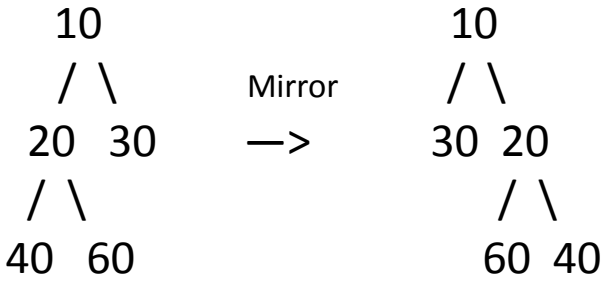
1. Mirror Tree:

Given a binary tree, convert it into its mirror.

Sample Input:



Sample Output:

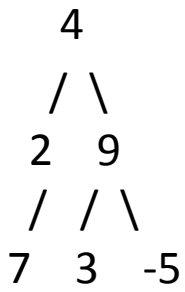


Inorder Traversal of mirror: 30 10 60 20 40

2. Level Min:

Given a binary tree, find the smallest value in each level.

Sample Input: [You can use dictionary here]



Sample Output: 4 2 -5

Explanation:

There are 3 levels in the tree

Level 0: {4}, min= 4

Level 1: {2, 9}, min= 2

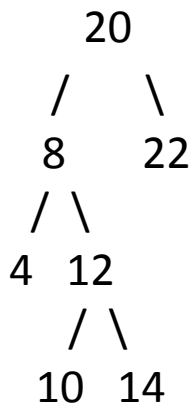
Level 2: {7, 3, -5}, min = -5

3. Inorder Predecessor:

Given a BST, and a reference to a Node x in the BST, find the Inorder Predecessor of the given node in the BST.

DO NOT USE LIST

Sample Input:



x = reference of the node containing 20

Sample Output: reference of the node 14

Explanation:

The inorder predecessor of a parent node is the largest (rightmost) node in the left subtree. The rightmost node in the left subtree of parent node 20 is 14.

Another explanation is that, the inorder traversal of the given tree:

4 8 10 12 14 20 22

Hence, the inorder successor of 20 is 14.

Sample Input 2:

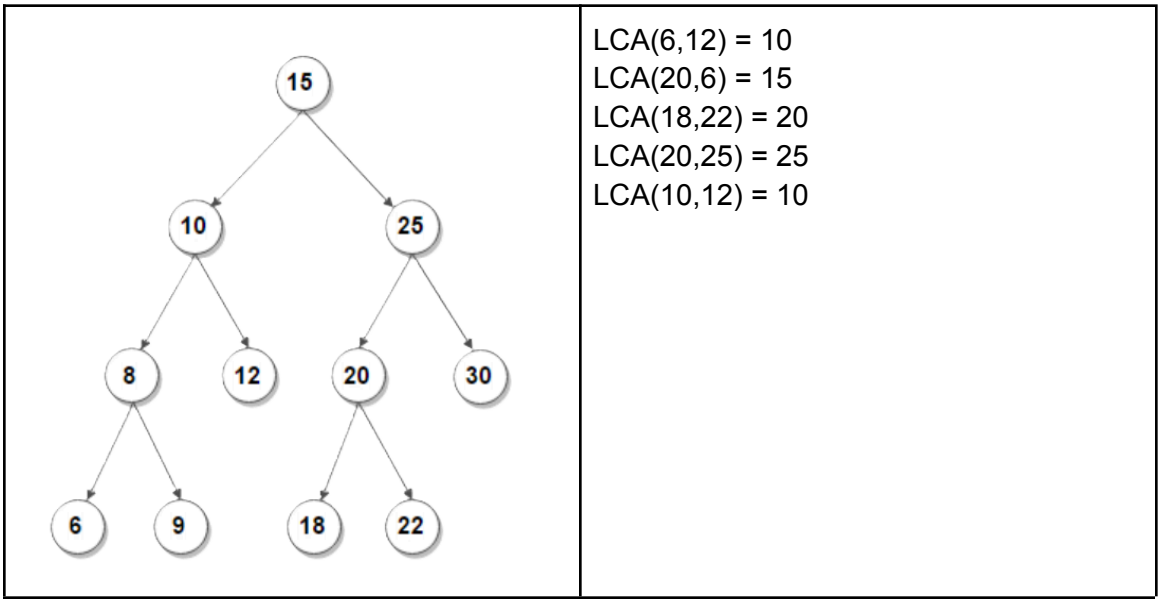
x = reference of the node containing 8

Sample Output: reference of the node 4

4. Find the Lowest Common Ancestor (LCA) of two nodes in a BST

The lowest common ancestor (LCA) of two nodes x and y in the BST is the lowest (i.e., deepest) node that has both x and y as descendants. In other words, the LCA of x and y is the shared ancestor of x and y that is located farthest from the root.

Corner Case: if y lies in the subtree rooted at node x, then x is the LCA; otherwise, if x lies in the subtree rooted at node y, then y is the LCA.



LCA(6,12) = 10
LCA(20,6) = 15
LCA(18,22) = 20
LCA(20,25) = 25
LCA(10,12) = 10

5. Sum of Nodes:

Alice and Bob are having a competition on who is better at math between them. To decide who is better, Alice gave Bob a task of summation. Alice gave Bob a tree and asked him to sum the values each node consist. However, Alice added a condition with this task. Bob needs to divide the values of each node with the node's level and sum the modulus except for root node. For root node, he needs to sum the value of the node. Now, Bob finds it difficult to do it but he can not loose. Thus, he came to you for help. Your task is to write a method that takes the root of a tree and return the sum.

*** You can create as much helper function you want**

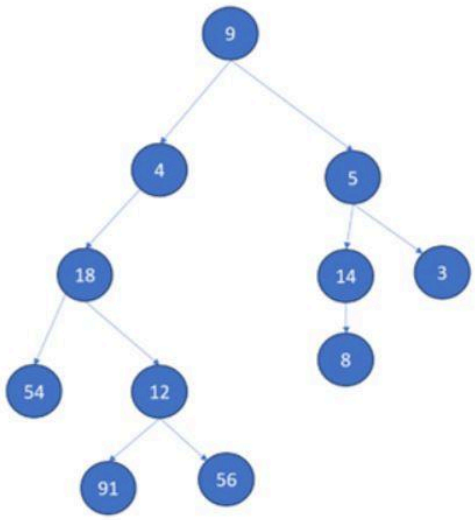
*** You can not modify the given codes nor the parameters.**

*** Hint: You know _order traversal of the tree.**

Example:

Test 1

Input



Output

15

Explanation:

Level 0 → 9

Level 1 → (4%1) + (5%1) = 0

Level 2 → (18%2) + (14%2) + (3%2) = 1

Level 3 → (54%3) + (12%3) + (8%3) = 2

Level 4 → (91 % 4) + (56%4) = 3

Output → 9+0+1+2+3 = 15

6. Swap Children Nodes:

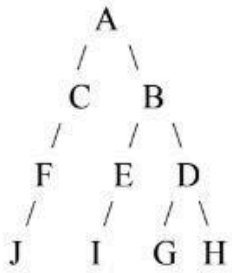
Write a **recursive** function **swap_child()** that takes the root of a binary tree, node's level and a number M as a parameter. The function will swap the left and right children of all the nodes at level M and above. Here, 0 < M < height of the tree (root's height). Consider, the Node class for Binary Tree already defined with elem, left and right variables. **YOU CANNOT USE LIST OR DICTIONARY, any built-in function, global variables.**

Python Notation:

```
def swap_child(root, level, M):
        # To do
```

Function Call :

swap_child(root, 0, 2). Here root refers to the tree below.

Input Tree	Resulting Tree	Explanation
		Here M = 2 and all the nodes from level 2 and above are swapped left with right. Here above means the level that situated at a higher position of the tree

7. Subtraction of Nodes:

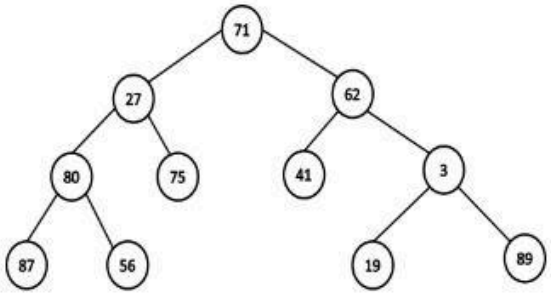
Write a **recursive** function **subtract_summation**() that takes the root of a binary tree as a parameter. The function will **subtract** the **summation** of the **right subtree** of the given root **from** the **summation** of the **left subtree** of the given root. Consider, the **Node** class for Binary Tree already defined with elem, left and right variables. You can use helper functions.
YOU CANNOT USE LIST OR DICTIONARY. You cannot use any built-in function.

Python Notation:

```
def subtract_summation(root):
    // To do
    return None
```

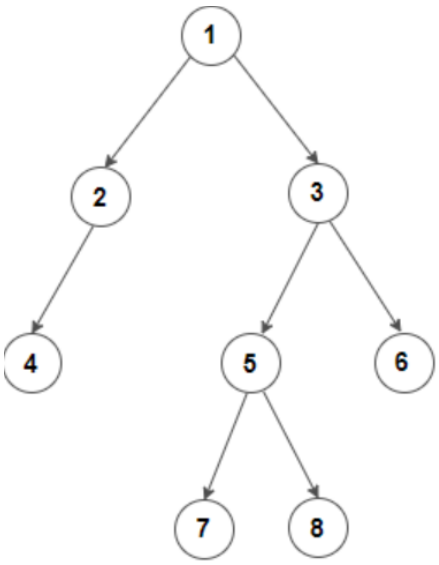
Function Call :

print(**subtract_summation**(root)). Here root refers to the tree below.

Sample Input	Sample Output	Explanation
	111	Summation of left subtree - summation of right subtree = (27+75+80+87+56) - (62+41+3+19+89) = 111

8. Bonus Task: Difference of Level Sum

Given a Binary Tree, Write a function that finds the difference between sum of all nodes present at odd and even levels in a binary tree, i.e. sum of all odd level nodes - sum of all even level nodes.

Sample Input:	Sample Output	Explanation
	4	-1+2+3-4-5-6+7+8 = 4