LAB 5 : Binary Tree

[CO2]

Instructions for students:

- Complete the following methods.
- You may use Java / Python to complete the tasks.
- DO NOT CREATE a separate folder for each task just follow the given template.
- If you are using **JAVA**, then follow the **Java Template**.
- If you are using **PYTHON**, then follow the **Python Template**.

NOTE:

- **YOU CANNOT USE ANY OTHER DATA STRUCTURE OTHER THAN ARRAY UNLESS MENTIONED IN THE QUESTION.**
- **YOUR CODE SHOULD WORK FOR ANY VALID INPUTS.**

Python List, Negative indexing and append() is STRICTLY prohibited

Lab Tasks: 1-3 (3 tasks)
Assignment Tasks: 4-6 (3 tasks)
Total Assignment Mark: 3*5=15

Lab Task 0: Basics of Binary Tree

This is an intro task & there isn't any driver code for this

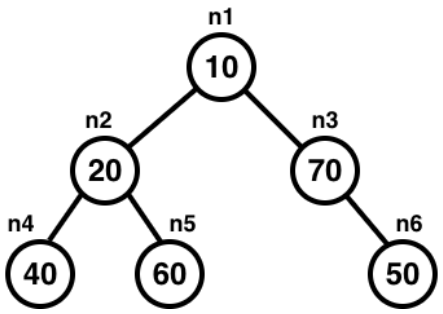
- ❖ For java, create a separate folder for this task and follow the instructions.
- ❖ For Python, create a separate colab/ipynb/py file and follow the instructions.

➔Design a TreeNode class.

- ◆ Declare three instance variables, one called **elem** (Object data type for java), one called **left** (TreeNode datatype) and another called **right** (TreeNode datatype).
- ◆ Write a constructor that has only one parameter (Object type for Java) and assigns the parameter value to the **elem** instance variable.

➔Design a Tester class (for java) / Design another code cell (for python)

- ◆ Create 6 different objects of TreeNode class. Assign values as shown in the illustration.
- ◆ Variable names should be: **n1, n2, n3, n4, n5, n6**.
- ◆ Connect the 6 TreeNodes as shown in the illustration which will form a binary tree. Here **n1** will be the root of the tree.



Now, execute the lines given below and try to understand the output. You may need pen & paper. If there are errors, try to figure out why that error occurred and how to fix it.

Note: Java & Python output might not always be the same.

JAVA	PYTHON
`System.out.println( n1.left );`	`print( n1.left )`
`System.out.println( n3.right.elem );`	`print( n3.right.elem )`
`TreeNode x = n2.left;` `System.out.println( n1.elem + x.elem );`	`x = n2.left` `print( x.elem + n6.elem )`

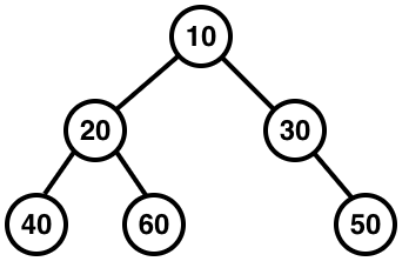
`x = new TreeNode(80);`
`n3.left = x;`
`System.out.println(n1.right.left.elem);`
	`x = TreeNode(80)`
	`n3.left = x`
	`print(n1.right.left.elem)`

`System.out.println(n1.left.right + n5.left);`	`print(n1.left.right + n5.left)`
`n1.left.right = null;` `System.out.println(n1.left.right.elem)`	`n1.left.right = None;` `print(n1.left.right.elem)`

1. InOrder Traversal **[LAB TASK]**:

Given the **root** of a binary tree, print the tree in-order.

Sample Input:



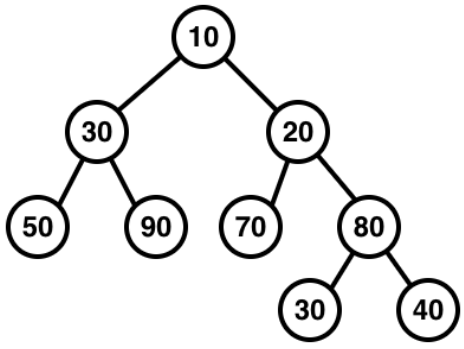
Sample Output:

InOrder Traversal : **40, 20, 60, 10, 30, 50**

2. Count the Total Nodes in a Binary Tree **[LAB TASK]**:

Given the **root** of a binary tree, find the total number of nodes in that tree.

Sample Input:

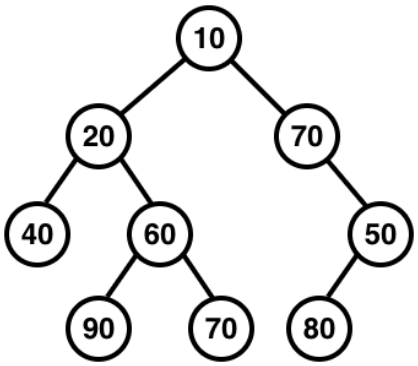


Sample Output:

Number of Nodes: **9**

3. Print Kth Level of a Binary Tree **[LAB TASK]**:
   Given the **root** of a binary tree and an integer K, print all the nodes that appear at the K-th level of the tree.

   Sample Input:



   K = 2

   Sample Output:

   Level 2 Nodes: **40, 60, 50**

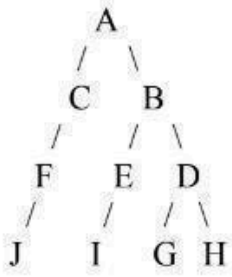
4. Swap Children Nodes **[ASSIGNMENT TASK]**:

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

5. Subtraction of Nodes **[ASSIGNMENT TASK]**:

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.

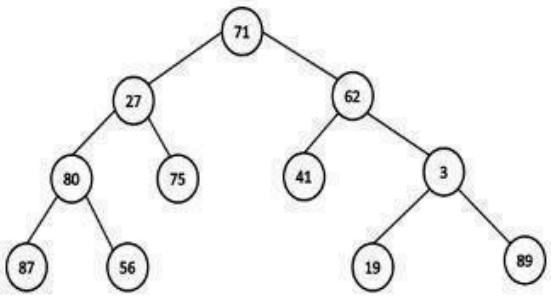
YOU CANNOT USE LIST OR DICTIONARY. You cannot use any built-in function.

Python Notation:

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

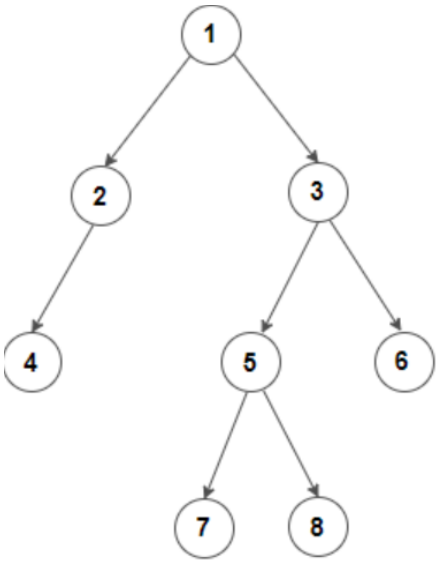
Function Call :

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

6. Difference of Level Sum **[ASSIGNMENT TASK]:**

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