

BST [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.

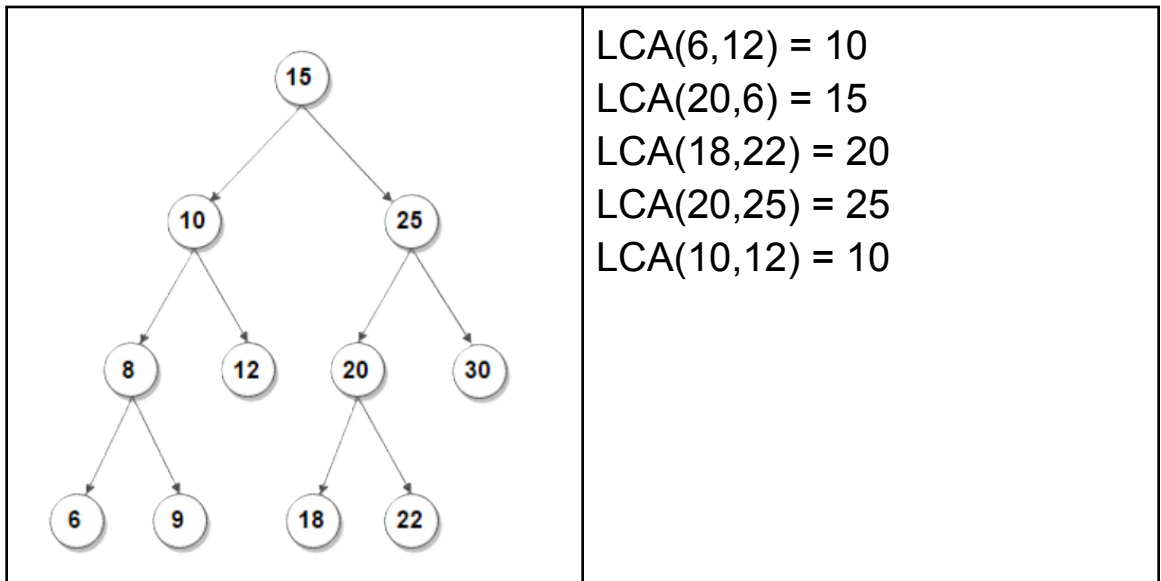
NOTE:

- **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**

1. 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.

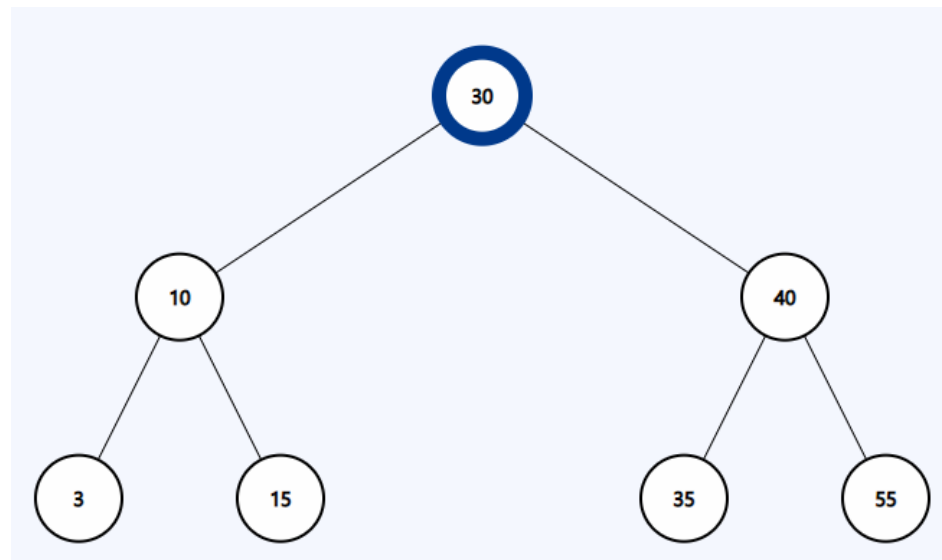
Corner Case: if y lies in the subtree rooted at node x , then x is the LCA; otherwise, if x lies in the subtree rooted at node y , then y is the LCA.



2. Finding a Path from source to destination in BST

Faria studies in Rangpur Medical College. During her Eid vacation, she is planning to visit her family and relatives in Chittagong. But she feels really bad during the journey. As a result, you need to help Faria reach her destination.

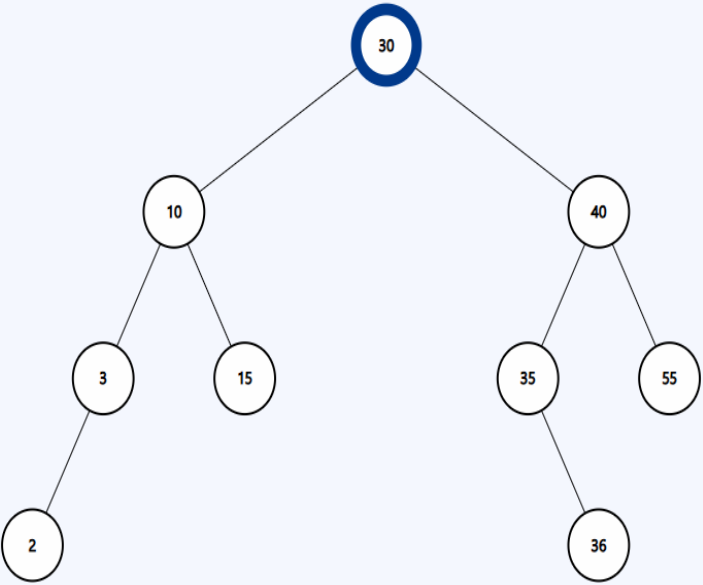
A Binary Search Tree is given. The root node is the starting position of Faria and a key node will be given as the destination. Now, you have to find if the following key node (which is the destination of Faria) is present in the tree or not. If present, return the path from the root node to the key node else print “No Path Found”.



Sample Input	Sample Output
Source node(root): 30 Destination node(key): 15	Path: [30, 10, 15]
Source node(root): 30 Destination node(key): 50	No Path Found

3. Sum of all Leaf nodes

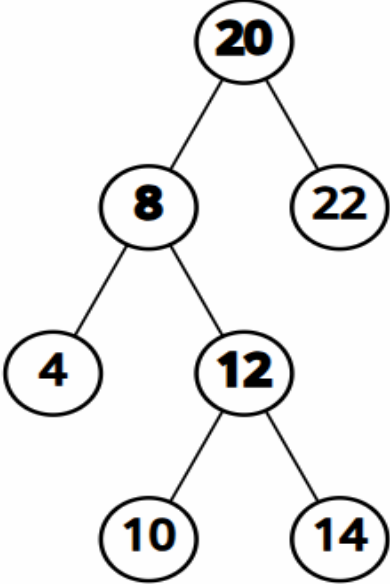
Write a function **sum_of_leaves(root)** that takes the root of a binary search tree as input and prints the sum of all the leaf nodes of the tree.

Sample Input	Sample Output
	<p>Sum of all leaves=2+15+36+55=108</p>

4. Inorder Predecessor

Write a function/method `in_order_predecessor(root, x)` that takes the root of a Binary Search Tree and another node `x` of the same tree in its parameter. Find the **Inorder Predecessor node** of the given node `x` from the BST and **return that node**.

NOTE: DO NOT USE LIST

Sample Input	Sample Output 1
	<p>Inorder predecessor of node 20: 14</p> <p>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.</p>
	<p>Sample Output 2</p> <p>Inorder predecessor of node 10: 8</p> <p>Explanation: If there are no left subtree then by the inorder traversal of the given tree: 4 8 10 12 14 20 22 We can say the inorder successor of 10 is 8.</p>