1. Write a function that will print nodes for each level of a binary search tree (Hint use a queue): : Example

10

/ \

5 13

/ \ \

3 7 14

The output of the function will be:

10

5 13

3 7 14

1. Write a function that will print the largest element in each level of a binary search tree.

10

/ \

5 13

/ \

3 7

The output of the given tree when the function is applied:

10

13

7

1. Check whether the leaf nodes of a BST is even or odd.

10

/ \

5 13

/ \ \

3 7 14

The output of the BST above is [odd, odd, even].

1. Traverse a BST and store in the numOfDescendants variable, the number of descendants that each of the nodes in the BST contains.

Graphical user interface, diagram, application

Description automatically generated

For example in the above figure: Output will be as follows:

7 has 5 descendants

4 has 3 descendants

8 has 0 descendant

3 has 0 descendant

6 has 1 descendant

1. has 0 descendant
2. Check whether two BSTs are identical or not. Write a function isDuplicate(root1, root2) that will check whether the BST with root1 is exactly identical to the other BST with root2.
3. Write a function int findLCA(root, nodeA, nodeB) that will return the value of the lowest common ancestor of the given nodeA and nodeB.

Detail Hints: https:/[/www.geeksf](http://www.geeksforgeeks.org/lowest-common-ancestor-in-a-)o[rgeeks.org/lowest-common-ancestor-in-a-](http://www.geeksforgeeks.org/lowest-common-ancestor-in-a-) binary-search-tree/

A picture containing clock

Description automatically generatedExample:

LCA of 10 and 14 is 12

LCA of 14 and 8 is 8

LCA of 10 and 22 is 20

1. Write a Function bool isBST(root) to check whether the tree with **root** is a

valid BST or not. Please write the function iteratively and recursively.

1. Write a function int calculatesum(root) to calculate sum of all nodes of the BST with root. Please write the function iteratively and recursively.
2. Write a function int countNodes(root) to calculate number of all nodes of the BST with root. Please write the function iteratively and recursively.