1a. bool function(tree1, tree2)

Check tree1==null && tree2==null. Return true if they do

Check that tree1==tree2. return false if not

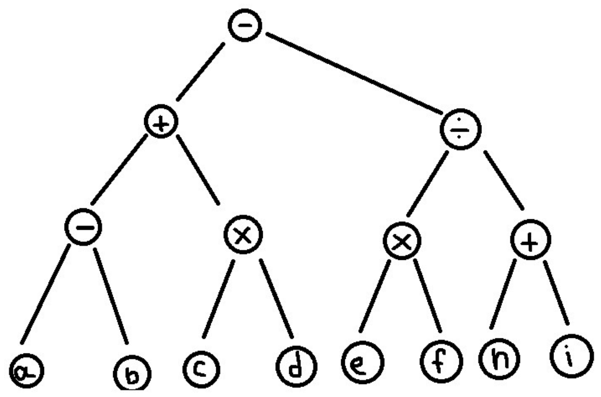
Recursively go left via function(tree1->left,tree2->left)

Recursively go right via function(tree1->right,tree2->right)

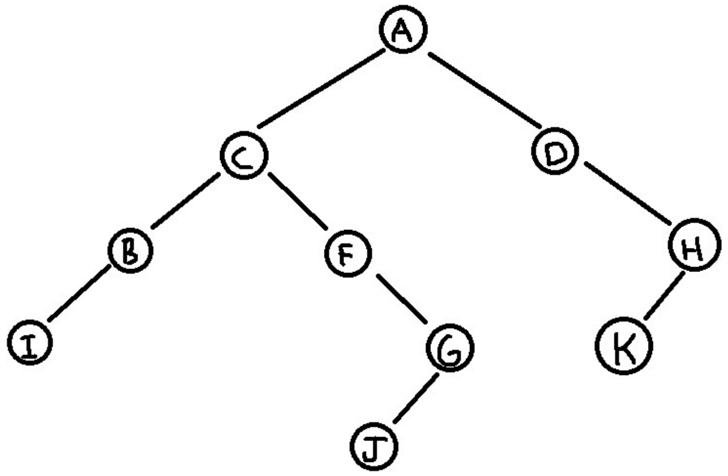
Return true

1b. The Big-O of the algorithm is O(n) as we are visiting each node once

2.



3.



4a. 4

4b. 0

4c. 1

4d.3

4e. 1, 20, 52, 83, 99, 125, 152

4f. 0

4g. Pre: 100, 50, 3, 1, 20,80,52,90,83,99,150,125,152

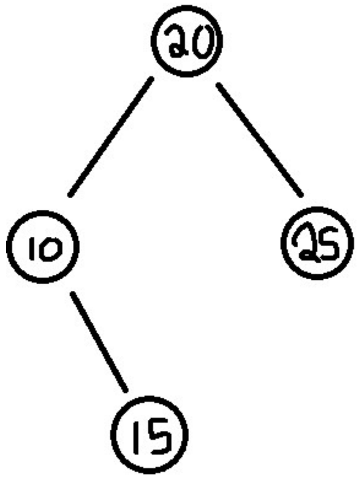
In-Order: 1,3,20,50,52,80,83,90,99,100,125,150,152

Post:1,20,3,52,83,99,90,80,50,125,152,150,100

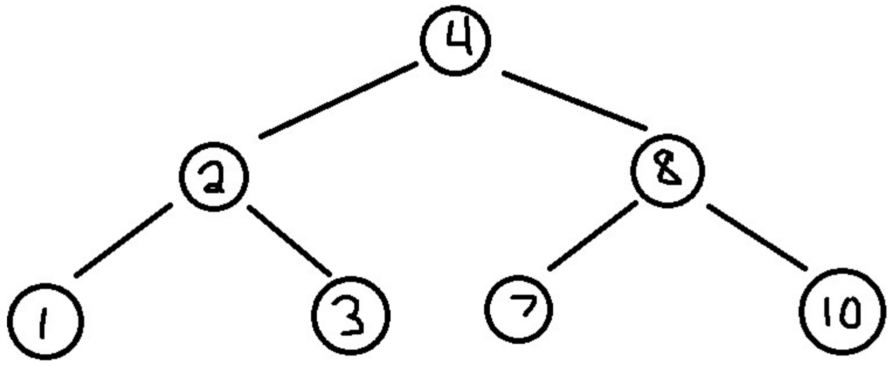
5a. A self-balancing BST where the difference of height between two subtrees of any given node cannot be more than one.

5b. To maximize the efficiency of a BST and ensuring that our big-O is as close to log(n) as possible

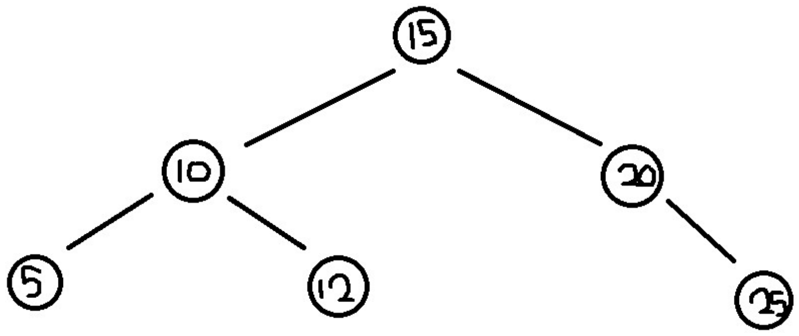
6.



7.



8.


15
10
20
5
12
25