1.

a. Create node class

   Initialize variables

   Create newNode class

   Initialize variables

   Return Node

   Create integer function

   Arguments: first tree, second tree

   If first tree is empty and second tree is empty

   Then return true

   If first tree is not empty and second tree is not empty

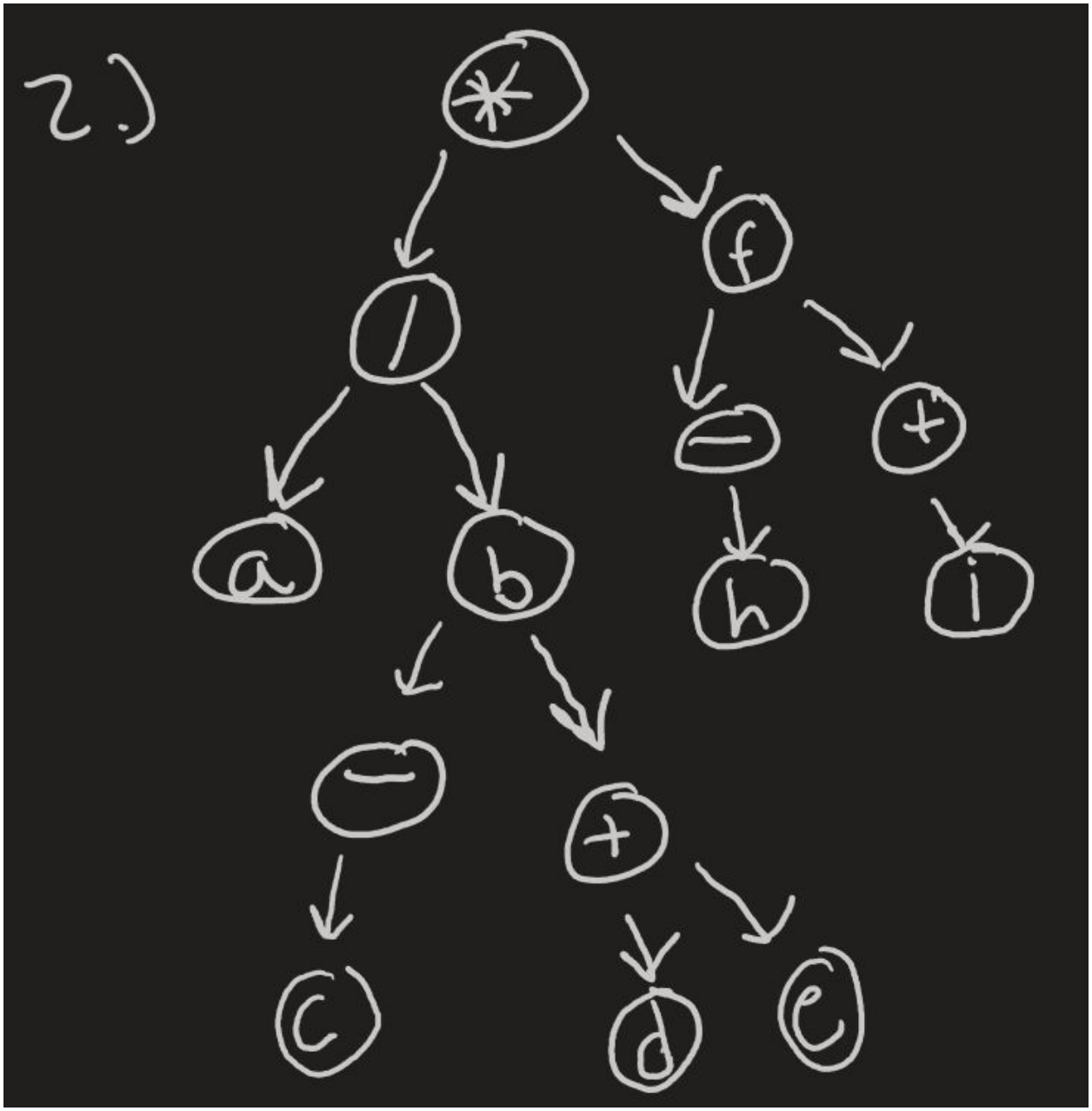
   Then check if the trees are the same

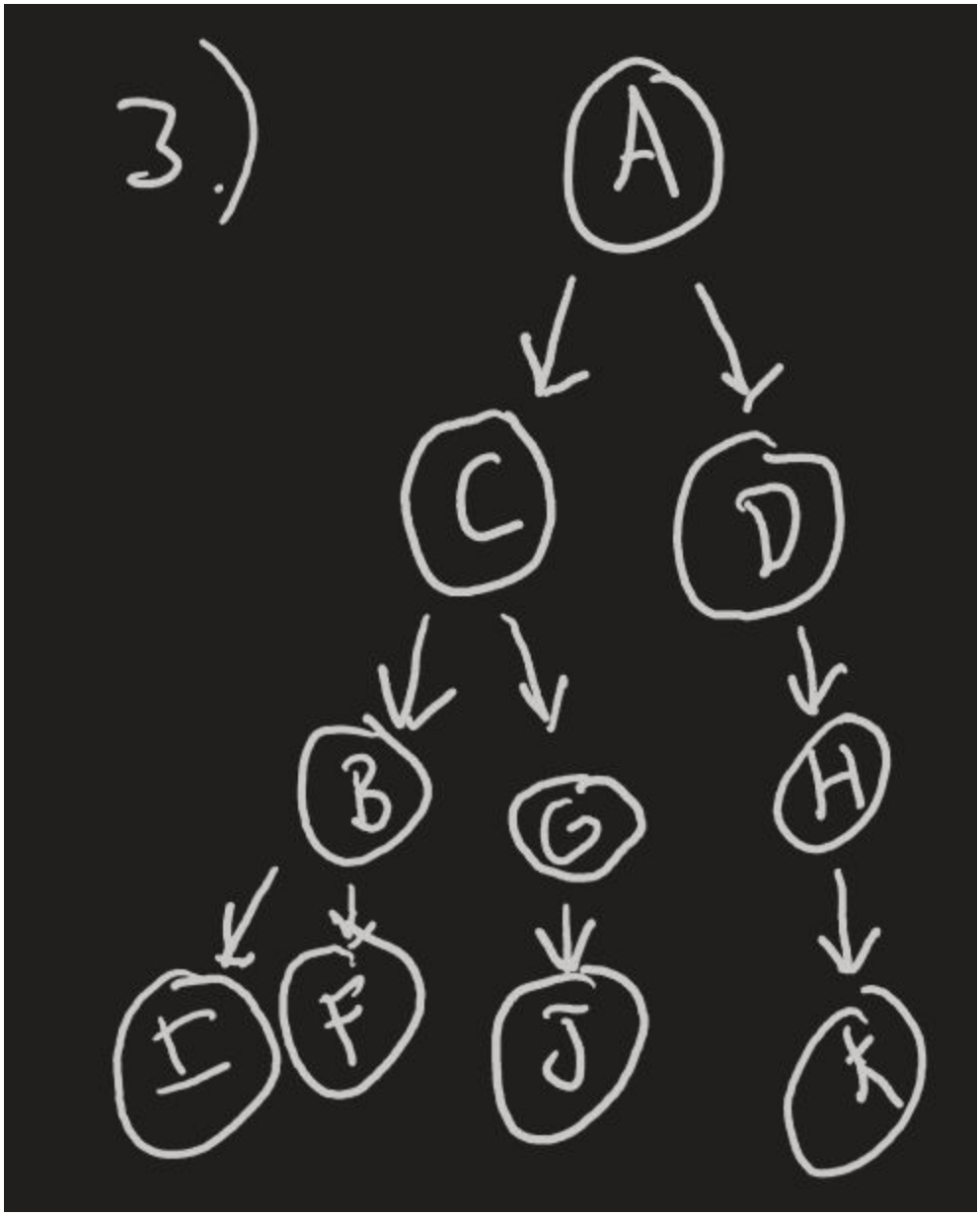
   Return true if they are the same

   Return false if they are not the same

   End

b. Worst case: O(n)


2.)
*
/
f
=
+
a
b
h
i
-
+
c
d
e



4.

a. Height = 4

b. Depth of root node = 1

c. Level of root node = 1

d. Depth of node 0125 = 2

e. Values of leaf nodes: 0001, 0020, 0052, 0083, 0099 (Not sure if those are 9's or not but the other branching off of 0090 next to 0083), 0125, 0152

f. Height of node 00125 = 0

g. Pre-order: 0100,0050,0003,0001,0020,0080,0052,0090,0083,0099,0150,0125,0152

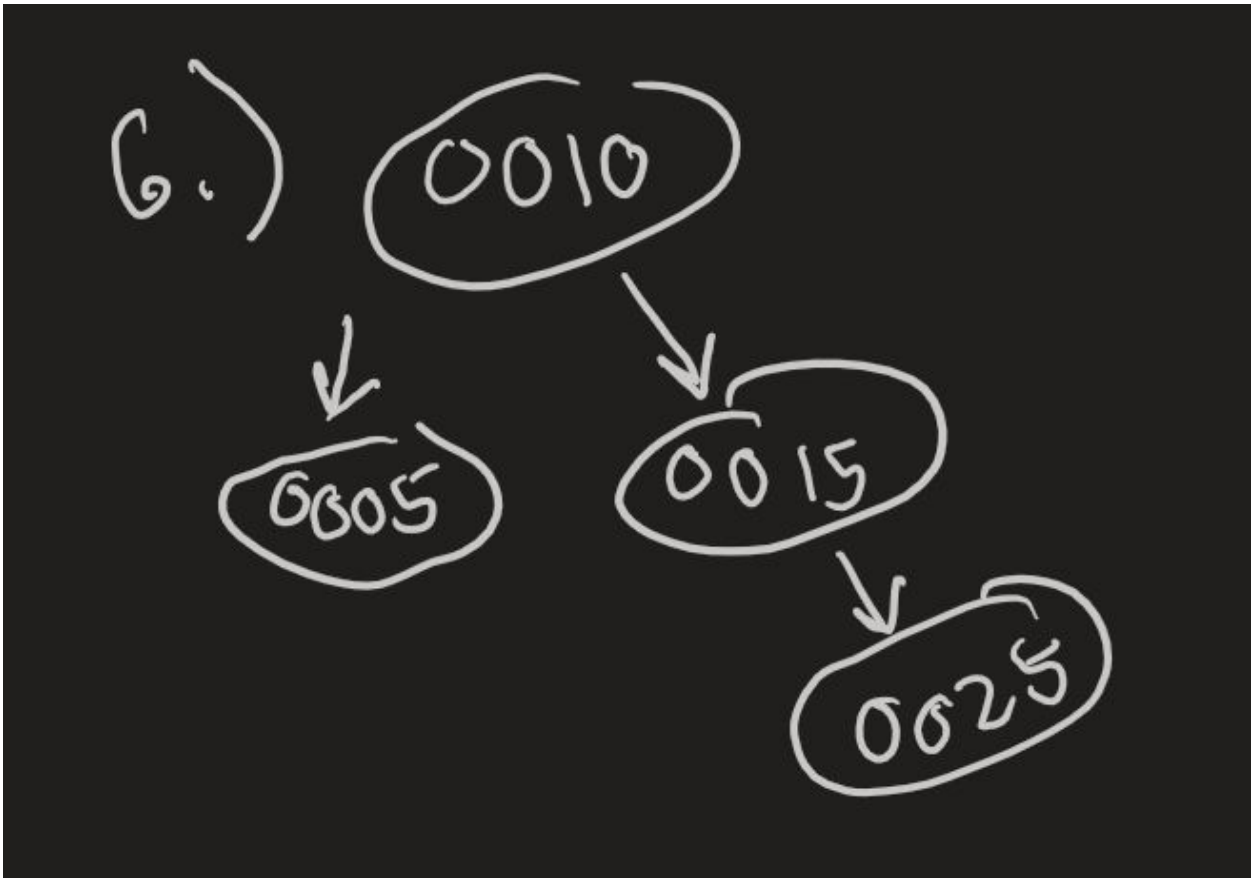
In-order: 0001,0003,0020,0050,0080,0052,0090,0083,0099,0100,0125,0150,0152

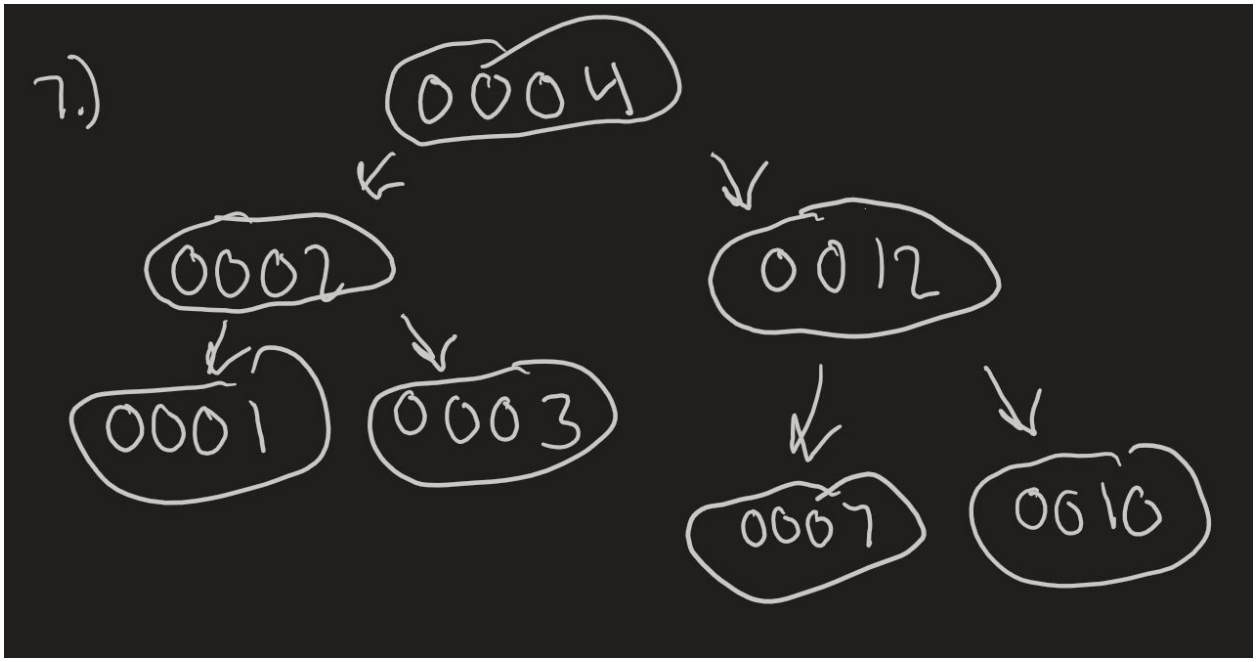
Post-order: 0001,0003,020,0083,0090,0080,0050,0125,0150,0152,0100

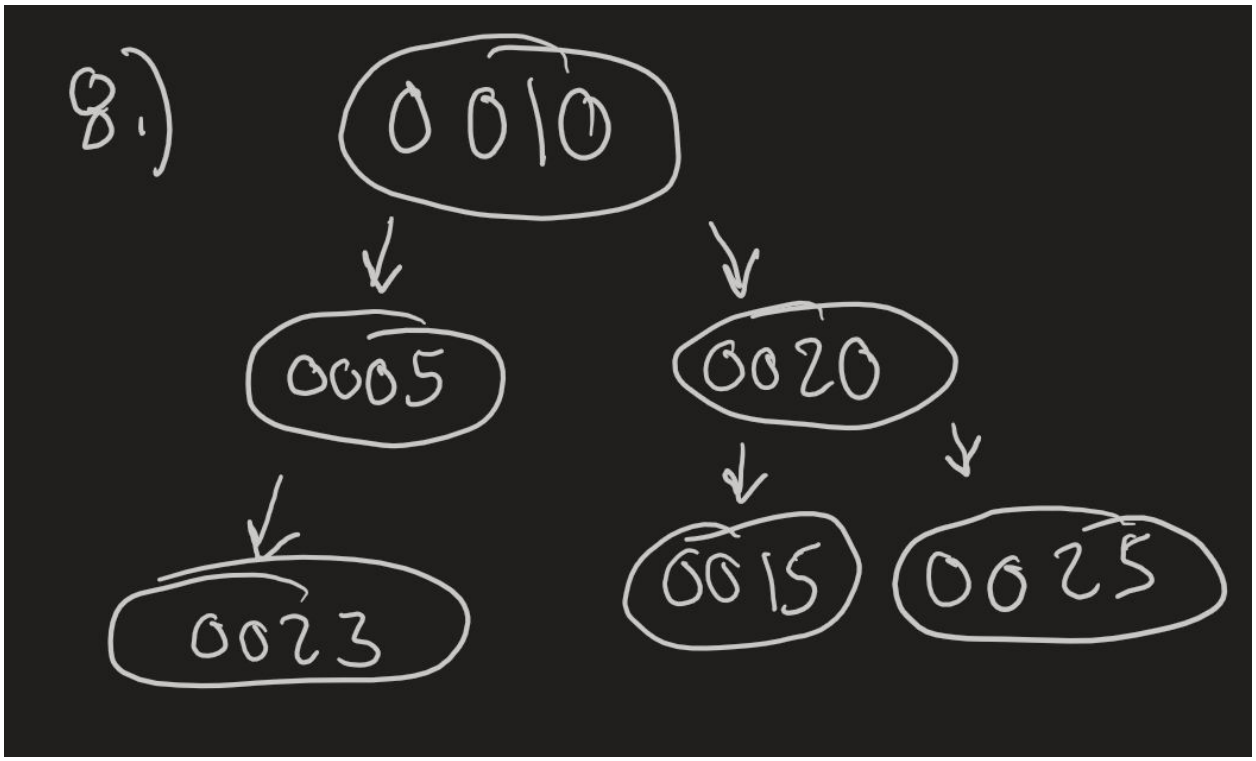
5.

a. An AVL tree is a self-balancing BST. It means it balances the tree by making sure that the height between each subtree is at most one.

b. The purpose of an AVL tree is to get the maximum potential of a binary tree. Since it self balances it means there are only two children for each node. This makes the binary tree get its maximum potential




7.)
0004
0002
0012
0001
0003
0007
0010


8.)
0010
0005
0020
0023
0015
0025