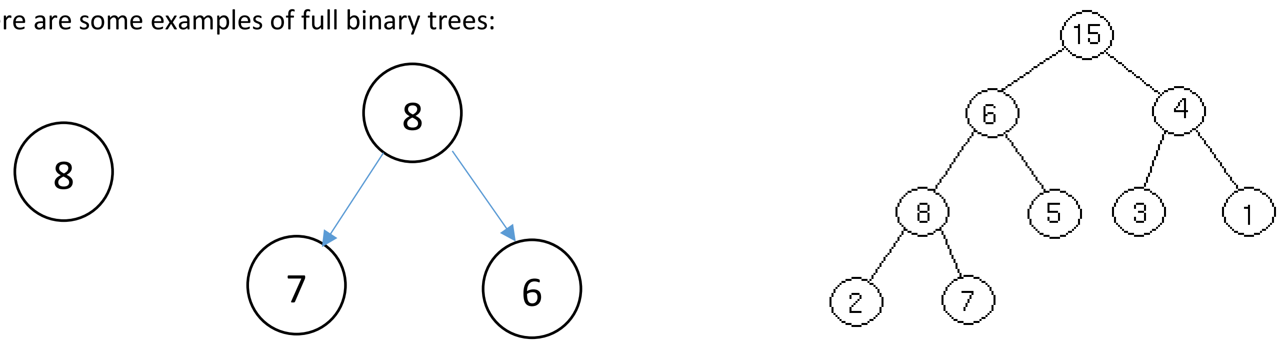
ITCS 209 Object Oriented Programming	Name:	Lab Score	Challenge Bonus	Peer Bonus
	ID:			
	Sec:			

Lab12: Recursion

A full binary tree is a tree in which every node other than the leaves has two children. Recursively, a full binary tree is either:

1. An empty tree (i.e. NULL)
2. A single non-empty (i.e. not NULL) node.
3. A graph formed by adding a non-empty full binary tree to left and a non-empty full binary tree to the right children of a non-empty node.

Here are some examples of full binary trees:



You are given `Node` class that implements a basic node that supports binary tree structure, and `FullBinaryTreeTester` class whose couple of methods are left blank for you to fill in. Specifically, you need to implement the following methods:

`public static void` `inOrderTraverse(Node root)`: Recursively traverse the tree from the root in an in-order fashion, and print out the node id visited.

`public static boolean` `isFullBinTree(Node root)`: Reclusively check the tree specified by "root" whether it is a full binary tree. Return true if it is, false otherwise.

Additionally, you may add any data structures and/or methods to facilitate your code.

Expected output from `normalTester()`:

```
[T0] in-order:
[T0] is a full binary tree.

[T1] in-order: 16
[T1] is a full binary tree.

[T2] in-order: 14 16
[T2] is NOT a full binary tree.

[T3] in-order: 6 3 7 1 8 4 10
[T3] is a full binary tree.

[T4] in-order: 3 1 8 4 10
[T4] is a full binary tree.

[T5] in-order: 6 3 1 8 4 10
[T5] is NOT a full binary tree.

[T6] in-order: 6 3 7 1
[T6] is NOT a full binary tree.
```

Challenge Bonus (Optional):

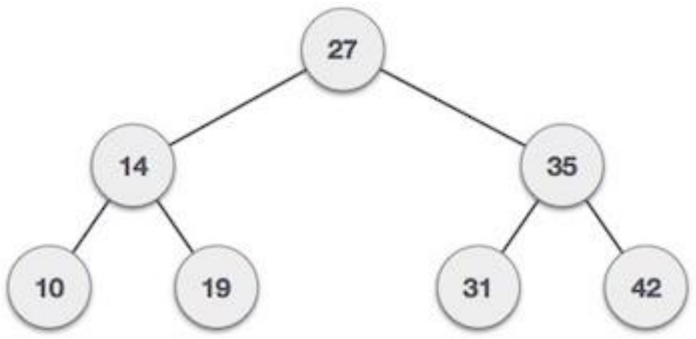
Implement the following methods.

`public static void printBinTree(Node root)`: Visualize (by printing out on console) the tree specified by "root." You do not need to stick with the example format below. Your code should work on any binary tree (i.e. even though it is not a full binary tree).

`public static Node getBinSearchTree(Node root)`: Transform a given binary tree into a *binary search tree*. A binary search tree (i.e. BST) is a tree in which all nodes have the following properties:

1. The left sub-tree of a node has value less than or equal to its parent node's value.
2. The right sub-tree of a node has value greater than or equal to its parent node's value.

For example,



Note that there can be multiple legit versions of BSTs given a set of nodes. Hence, your output may appear different from the sample output below. At this point, just make sure your output is a BST. Your tree formatting does not need to be the same as the sample output.

Sample output from `bonusTester()`:

```
Before Transforming:
        1
      /   \
    3       4
   / \     / \
  6    7  8    10
After Transforming:
        6
      /   \
    3      8
   / \    / \
  1   4  7    10
```

Peer Bonus (Optional):

Sticker Here

Sticker Here

Sticker Here