# CS 61B Exam Prep 9: Trees Spring 2020

1 Binary Trees

For each of the following questions, use the following `Tree` class for reference -

```java
public class Tree {
    public Tree(Tree left, int value, Tree right) {
        _left = left;
        _value = value;
        _right = right;
    }
    public Tree(int value) {
        this(null, value, null);
    }
    public int value() {
        return _value;
    }
    public Tree leftChild() {
        return _left;
    }
    public Tree rightChild() {
        return _right;
    }
    private int _value;
    private Tree _left, _right;
}
```

(a) Given a binary tree, check if it is a sum tree or not. In a sum tree, value at each non-leaf node is equal to the sum of all elements presents in its left and right subtree. For example, the following binary tree is a sum tree -
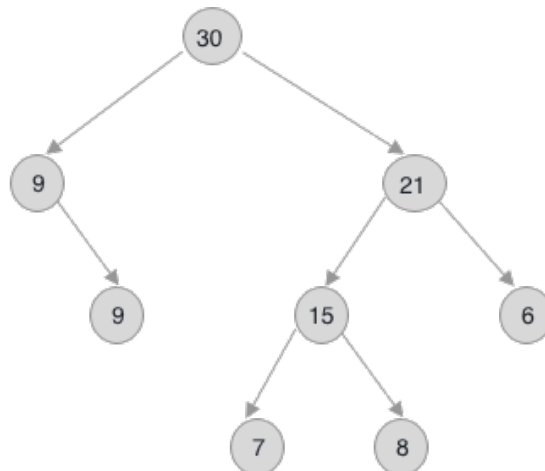
```
public boolean isSumTree(Tree t) {
    _____
    _____
    _____

    _____
    _____
    _____
    _____
    _____
    _____
}
```

(b) Given a binary tree with distinct items, an input list, and an empty output list, add all the elements in the input list to the output list that appear in the tree. The elements in the output list should be ordered in the same order that would be returned from an inorder traversal.

For example, for the tree in Q.1(a) assuming it has only distinct items, if the input list is [15, 9, 8, 30, 6], then after the operation the output list should be [9, 30, 15, 8, 6].

```
public static void sortRelative(Tree t,
                                List<Integer> inputList,
                                List<Integer> outputList) {
    _____
    _____
    _____
    _____
    _____
    _____
    _____
}
```

(c) Given a Tree t and two integers x and y, return the lowest common ancestor of x and y in t. Assume all labels in t are nonnegative, nonzero, and distinct, and that x and y are in t. The behavior of the function can be undefined if the above does not hold.

For example, for the tree in Q.1(a), the lowest common ancestor for node 6 and node 7 will be node 21.
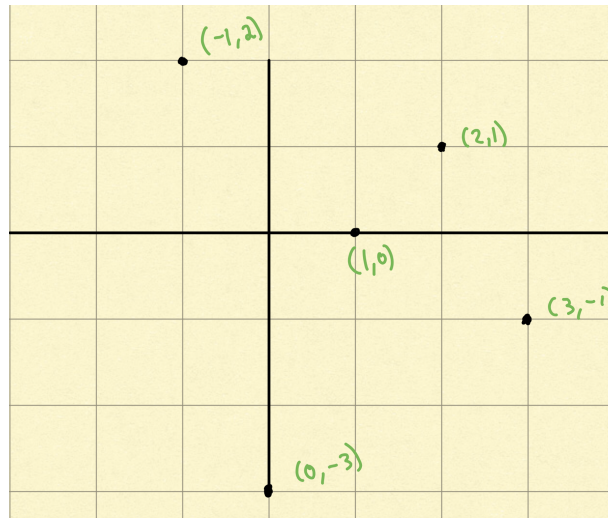
```
public static int commonAncestor(Tree t, int x, int y) {
    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____
    _____
}
```

2 Quad Trees

Assuming we are starting with an empty quad tree, using the points below, what order of insertion produces the quad tree with -

(a) Maximum Height

(b) Minimum Height

3 Challenge

Given five points along the line y = x and an initially empty quad tree, how many different orders of insertion produce a quad tree of maximum height?

Now instead of five points, what if we were given n points?