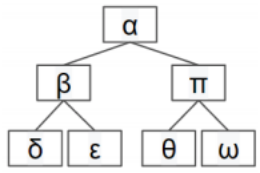
CS 61B Spring 2018

Heaps and Trees

Exam Prep 10: March 19, 2018

1 An Operational Understanding (Spring 2017 MT2 Q2)

Consider the tree on the left where greek letters represent numerical values. In the boxes to the right, shade all values that might match the text. Assume all values are unique. For BSTs, assume left items are less than.



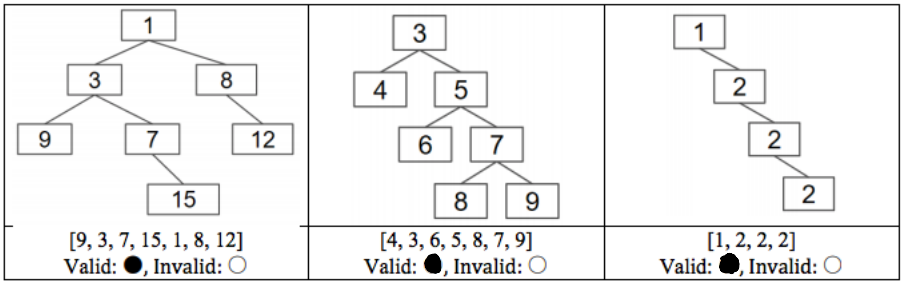
MinHeap, largest item	☐α	☐β	☐π	☑δ	☑ε	☑θ	☑ω	
MinHeap, smallest item	☑α	☐β	☐π	☐δ	☐ε	☐θ	☐ω	
BST, largest item	☐α	☐β	☐π	☐δ	☐ε	☐θ	☑ω	
BST, smallest item	☐α	☐β	☐π	☑δ	☐ε	☐θ	☐ω	
MinHeap, median item	☐α	☑β	☑π	☑δ	☑ε	☑θ	☑ω	
BST, median item	☑α	☐β	☐π	☐δ	☐ε	☐θ	☐ω	
MinHeap, new root after deleteMin	☐α	☑β	☑π	☐δ	☐ε	☐θ	☐ω	
BST, new root after Hibbard[1] deletion of α	☐α	☐β	☐π	☐δ	☑ε	☑θ	☐ω	
MinHeap, root after inserting new item φ	☑α	☐β	☐π	☐δ	☐ε	☐θ	☐ω	☑φ
BST, root after inserting new item φ	☑α	☐β	☐π	☐δ	☐ε	☐θ	☐ω	☐φ

2 Xelha Trees (Spring 2017 MT2)

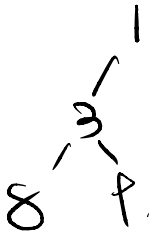
Given a list of numbers X, a XelhaTree for that list obeys the following:

1. The XelhaTree has the min-heap property (i.e. every value is less than or equal to its children).
2. An inorder traversal of the XelhaTree visits the nodes in the same order as the list.

(a) Which of the following are valid XelhaTrees for the given sequences? The first is done for you.



(b) Draw a valid XelhaTree corresponding to the sequence [8, 3, 9, 1].



3 Verifying Xelha Trees (Spring 2017 Final Q10)

Write a function `validXelhaTree` which takes an `IntTree` and a `List` and returns true if the `IntTree` is a `XelhaTree` for the list. You may not need all lines. A `XelhaTree` is valid if it obeys the min heap property, and if an in-order traversal of the `XelhaTree` yields the list of items passed to `createXelhaTree` (in the same order). One line if statements with on the same line are fine. You may not need all the blanks. Assume there are no duplicates.

```
1  public class TestXelhaTree {
2      public static class IntTree {
3          public int item;
4          public IntTree left, right;
5      }
6
7      public static IntTree createXelhaTree(List<Integer> x) { ... }
8
9      /** If x is null, returns largest possible integer 2147483647 */
10     private static int getItem(IntTree x) {
11         if (x == null) { return Integer.MAX_VALUE; }
12         return x.item;
13     }
14
15     public static boolean isAHeap(IntTree xt) {
16     _____if(xt.left.getItem < Item && xt.right.getItem< Item){
17     _____  return false;}_____________________________
18     _____if(xt.left ≠ null){ isAHeap(xt.left);}.
19     _____if(xt.right≠null){ isAHeap(xt.right);}.
20     _____return true;_____________________________
21     _____________________________________________________ }
22
23     public static void getTreeValues(IntTree xt, List<Integer> treeValues) {
24     _____if(xt.left!null){ getTreeValues(xt.left, treevalues);}.
25     _____treevalues.add(xt.item);_____
26     _____if(xt.right!null){ getTreeValues(xt.right, treevalues);}.
27     ____________________________________ _____________
28     _____________________________________________________
29     _____________________________________________________ }
30
31     public static boolean validXelhaTree(IntTree xt, List<Integer> vals) {
32         List<Integer> treeValues = new ArrayList<Integer>();
33         /* getTreeValues adds all values in xt to treeValues */
34         getTreeValues(xt, treeValues);
35         _____________________________________________________
36         _____________________________________________________
37         return isAHeap(xt) && ___vals.equals(treeValues);.
38 }
```

if(xt=null){return true;}.

if(xt=null){return}.

4 Reconstructing Trees from Traversals

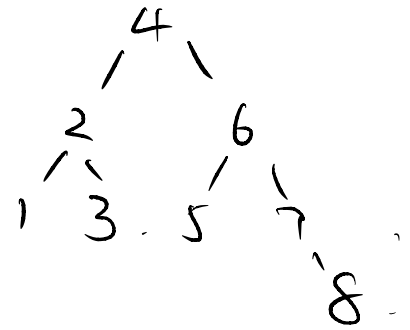
Given two lists of integers, where one represents the preorder traversal of a binary tree, and the other represents the inorder traversal of a binary tree, come up with a high-level implementation to reconstruct a binary tree given this information. In other words, you are coming up with an implementation for the following method:

```
public IntTree constructTree(int[] preorder, int[] inorder)
```

You may assume that all the elements in the lists are distinct.

See Guy's solution for details

keys { preorder ⇒ know the root
inorder ⇒ know left & right



4 2 1 3 6 5 7 8.

1 2 3 4 5 6 7 8.