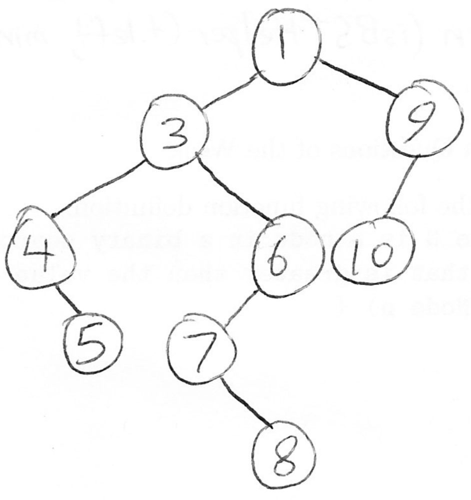
CS61B Week 8: Collections and Trees

1. The Collections you should know about by now are: HashSet, HashMap, ArrayList, LinkedList, Stack, Queue, and TreeSet. Out of these, which would you use for each of the following situations?
   (a) Creating a database so that you can quickly retrieve the name of a Berkeley student given their student ID. HashMap Key → SID Value → Name
   (b) Keeping track of who is logged on to one of the EECS instructional servers. Must be able to quickly check if someone with a given student ID is currently logged on. HashSet
   (c) Keeping track of which pieces are where on an expandable chess board. Probably HashMap of spaces to pieces. Maybe 2D ArrayList
   (d) Representing people in line at a store. Make sure you can handle people cutting, assuming you know who they're cutting in front of. LinkedList
   (e) Making a roll sheet of students in a class. The roll sheet should stay alphabetized and you shouldn't have problems when students add or drop the class. TreeSet (implements SortedSet)
   (f) Keeping track of the waitlist for the iPhone 5. Keep in mind that whoever ordered first should get the first iPhone. Queue (implemented as LinkedList in Java)
   (g) Model people getting on and off of an airplane. Those with seats at the back of the plane get on first and off last. Stack

For all problems below, we have the following definitions:

```
public class Node {
    public Node parent;
    public Node left;
    public Node right;
    public int value;
}
void printInOrder(Node t) {
    if (t  ==  null) return;
    printInOrder(t.left);
    System.out.println(t.value);
    printInOrder(t.right);
}
void printPreOrder(Node t) {
    if (t == null) return;
    System.out.println(t.value);
    printPreOrder(t.left);
    printPreOrder(t.right);
}
void printPostOrder(Node t) {
    if (t == null) return;
    printPostOrder(t.left);
    printPostOrder(t.right);
    System.out.println(t.value);
}
```



2. Draw the tree: (1 (3 (4 null (5 null null)) (6 (7 null (8 null null)) null)) (9 (10 null null) null)).
   In this notation, (v l r) is a tree whose value is v, left subtree is l, and right subtree is r.
   Now, assuming Node n is a pointer to the root of the tree (whose value is 1), write what would be printed by printInOrder(n), printPreOrder(n), and printPostOrder(n).

In-order: 4 5 3 7 8 6 1 10 9
Preorder: 1 3 4 5 6 7 8 9 10
Postorder: 5 4 8 7 6 3 10 9 1

3. Implement an iterative preorder (aka depth-first) traversal of a tree. How could you change it into a breadth-first traversal by making only a couple small changes?
```
public void printPreOrderIterative(Node t) {
    Stack<Integer> s = new Stack<Integer>();
    s.push(t);
    while(!s.isEmpty()) {
        Node n = s.pop();
        System.out.println(n.value);
        if (n.left != null) s.push(n.left);
        if (n.right != null) s.push(n.right);
    }
}
```
For breadth-first, change the Stack to a LinkedList (acting as a queue)

4. Complete the following function definition:
```
/** Returns true iff binary tree T is a binary search tree. */
boolean isSearchTree(Node t) {
    return isBSTHelper(t, Integer.MIN_VALUE, Integer.MAX_VALUE);
}
boolean isBSTHelper(Node t, int min, int max) {
    if (t == null) return true;
    if (t < min || t > max) return false;
    return (isBSTHelper(t.left, min, t.value) && isBSTHelper(t.right, t.value, max));
}
```

Sample Midterm Questions of the Week:

1. Complete the following function definition:
```
/** Assume N is a node in a binary search tree. Returns the least value in this
 *  tree that is greater than the value of N, or null if there is no such value. */
int next(Node n) {
```

2. Given a binary tree where all nodes have an additional unset field *next*, write a function to fill in the next fields of all nodes so that following them gives an inorder traversal of the tree.