COMPUTER SCIENCE 112 - SPRING 2012
MIDTERM EXAM

Name: ________________________________

CIRCLE your recitation section: 01 02 03 12 13 14

- Be sure your test has 4 questions.
- **DO NOT TEAR OFF THE SCRATCH PAGES OR REMOVE THE STAPLE.**
- Be sure to fill your name and circle your recitation time above, and your name in all subsequent pages where indicated.
- This is a CLOSED TEXT and CLOSED NOTES exam. You MAY NOT use calculators, cellphones, or any other electronic device during the exam.

Do not write below this line

Problem	Max	Score
-------	-----	-----
1 Sorted Linked Lists Difference	20	_____
2 Sorted Array Insertion	20	_____
3 Binary Search Tree	17	_____
4 BST Ranking	18	_____
TOTAL	75	_____

1. **Sorted Linked Lists Difference (20 pts, 15+5)**

Suppose you are given two **sorted** lists of integers. The difference of the first list with respect to the second is the set of all entries that are in the first list but not in the second. For instance, given these two sorted lists:

```
L1: 3 --> 5 --> 9 --> 15 --> 19 --> 25

L2: 5 --> 8 --> 15 --> 18
```

The difference (L1 minus L2) is:

```
L1: 3 --> 9 --> 19 --> 25
```

Complete the following method to implement the difference of two sorted lists. Assume neither list has any duplicate items. You can implement helper methods if needed, and you have the option of using recursion.

```
public class LLNode {
   public int info; public LLNode next;
   public LLNode(int data, LLNode ptr) { info = data; next = ptr; }
}

// returns a pointer to the first node of a NEW linked list which is
// is list1 difference list2; returns null if the difference is the empty list
// the input lists are NOT modified
public LLNode difference(LLNode list1, LLNode list2) {
    // COMPLETE THIS METHOD
```

(b) What is the worst case running time of your implementation, if the length of the two lists are m and n. Identify the basic unit time basic operations, count the number of times they are done, total the counts, and convert to big O. You will not get any credit without an adequate derivation, even if your answer is correct.

2. **Sorted Array Insertion (20 pts, 15+5)**

a) Implement the *fastest* possible algorithm to insert a new entry into a sorted (in ascending order) array of items. Duplicates are NOT allowed (throw an exception if a duplicate is attempted to be inserted.) After insertion, the array should still be in sorted order. You will get at most half the credit if your algorithm is not the fastest possible. (Fastest here refers to the real clock time, not big O).

```
// Inserts an item into a sorted array A, containing n entries, where n is
// strictly less than the length of the array. (There are more spaces in the
// array than entries.) Throws an IllegalArgumentException if item already exists.
// After the insertion, the array is still sorted.
public static <T extends Comparable<T>> void sortedInsert(T[] A, int n, T item) {
```

b) What is the *worst case* big O running time for your implementation? Identify the basic operations and show how they add up to the running time. (For any of the search algorithms done in class, you may assume its known running time without derivaton.) You will not get any credit without an adequate derivation, even if your answer is correct.

3. **Binary Search Tree (18 pts, 7+4+3+4)**

Given the following sequence of integers:

```
        6, 36, 15, 4, 45, 27, 20
```

a) Starting with an empty binary search tree, insert this sequence of integers one at a time into this tree. Show the tree after every insertion. Briefly explain your insertion algorithm - it should ensure that duplicates are NOT inserted.

For parts (b) and (c) “comparison” refers to comparison of data items.

b) How many comparisons *in all* did it take to build this tree following the algorithm your described in part (a)? Show your work.

c) What is the *average* number of comparisons for a successful search in this tree, assuming they are searched for with equal probabilities? Show your work.

(d) Suppose a BST stores some integers in the range 1 to 500. We will perform a search for the value 225, and keep a list of the values encountered on the search path through the BST. For each of the following sequences of values, say whether or not it is a possible search path. If yes, show the actual path (with branches), if not, explain why.

i) 500, 400, 200, 300, 230, 350, 225

ii) 500, 355, 150, 320, 200, 225

iii) 50, 100, 300, 40, 120

4. **BST Ranking (17 pts, 10+7)**

You are given a Binary Search Tree (BST) with nodes defined as follows:

```
public class BSTNode<T extends Comparable<T>> {
    T data;
    BSTNode<T> left, right;
    ...
}
```

a) Describe an algorithm to find the k-th smallest item in the BST. (k=1 means smallest, k=2 means second smallest, etc.) Your description must be precise enough to be able to translate into Java code. You can use additional data structure(s) if you like, but you may NOT change the `BSTNode` class itself. Credit depends both on correctness (50%) and efficiency (50%).

b) What is the **worst-case** running time of your algorithm. Clearly state the basic unit time operations, show how many times each operation is done, add up all these, and convert to big O.

SCRATCH PAGE

SCRATCH PAGE