COMS W3134 Data Structures in Java – Section 1

Midterm Exam, Spring 2017

NAME:_____

UNI:_____

There are **7** questions on this exam totaling 100 points. The exam is closed book and closed notes. No calculators or computers are allowed. You will have 75 minutes to complete this exam. Do not open this exam packet until instructed.

Print your name and UNI on the exam. Read and sign the academic honesty statement below. Next, write your name and UNI on the front cover of your blue book.

Place all answers in your blue book. Answers written on the exam sheet itself will not be graded. If you need more than one blue book, number your blue books and put your name and UNI on all of them.

At the end of the exam submit **BOTH** your blue book **AND** your signed exam sheet. Your exam will not be graded if we do not receive both pieces.

Academic Honesty Statement:

I certify that I have neither given nor received unauthorized help on this exam and that I did not use any notes, electronic devices, or other aids not specifically permitted. I will not discuss the content of this exam with anyone who is not taking the midterm at this time. I understand that any violation of this policy can result in an exam grade of zero and will be reported.

Signature:_____

1. (12 points total) Using induction, prove that the number of nodes, N, in a perfect binary tree is odd.

2. (16 points total) Run times (for big-O costs, provide as tight a bound as you can get):

    a. Give the **worst** case big-O cost for the following algorithms:

        i. (3 points) Generating a postfix expression from an expression tree that has a total of N nodes (inclusive of both operators and operands).

        ii. (3 points) Contains operation on an AVL tree.

        iii. (3 points) Recursive calculation of Fibonacci numbers (the non-memoized version)

        iv. (3 points) Pop from a stack implemented efficiently with a singly linked list.

    b. (4 points) List the answers in part a from fastest growth rate to slowest growth rate. If any have the same growth rate put them next each other in the list and circle them.

3. (12 points total) Assume you have an initially empty Stack and a string of N characters. A program performs N push operations of each of those characters in the order in which they appear in the string. After each push, it performs any number of pop operations, such that the total number of pops is N. Assume that the pop operation prints the value that came off. For example, you could have the string:

    LEMONS

    Then you perform the following sequence of pushes and pops

    push(L), push(E), push(M), pop(), pop(), pop(), push(O), pop(), push(N), pop(), push(S), pop()

    It would result in the following output string:

    MELONS

    Now, consider the string:

    RESCUED

For following two output strings, provide the sequence of stack operations that would generate it from the initial string of RESCUED. If the sequence cannot occur, explain why.

a. SECURED

b. REDUCES

4. (10 points total) You are given both the pre-order traversal and the in-order traversal for a unique binary tree. Draw the corresponding tree and write down its post-order traversal.

   Pre-order traversal: B A C E D F

   In-order traversal: C A E B D F

5. (17 points total) A full binary tree is a binary tree in which each interior node has exactly two children. Write a **recursive** Java method, *boolean isFull(TreeNode root)* that, given a reference to the root node of a binary tree, returns true if the binary tree is full. Assume a standard Binary TreeNode implementation with left child and right child references. (Note, since a single node or empty tree has no interior nodes, they should be considered full.)

6. (17 points total) Imagine using a doubly linked list to implement the queue ADT for values of type *int*. Skeleton code is provided below. Implement the *enqueue* and *dequeue* methods. Instead of using the methods of the List ADT, manipulate the nodes directly. You have access to the head and tails nodes directly. This doubly linked list uses sentinel nodes and you can assume that head and tail have already been initialized properly.

```
class LinkedListQueue {

   private static class Node {
      Node prev;
      Node next;
      int data;
   }

   Node head;
   Node tail;

   public enqueue(int x) { // write this};
   public int dequeue() { // write this};

}
```

7. (16 points total) Starting with an empty AVL tree, show the tree after inserting each of the following values sequentially: 2, 1, 6, 7, 8, 3, 5, 4. Be sure to also show all rotations performed (double rotations should show both single rotations involved).