Room 309 HAV SEAT:____________

COMS W3134 Data Structures in Java – Section 1

Midterm Exam, Spring 2018

NAME:____________________

UNI:______________________

SECTION (1 or 2):__________

YOU MUST SIT IN THE SEAT DESIGNATED AT THE TOP OF THE EXAM. Failure to do so may result in a failing grade.

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.

Place all answers in this booklet. You may use the blank back of pages if you need additional space.

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:__________________________

DO NOT WRITE ON THIS PAGE!

Failure to comply will result in… well you know.

1	2	3	4	5	6	7	Total

1. (12 points total) Using induction, prove that in a full binary tree with N interior nodes (non-leaves), there are N+1 leaf nodes.

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 a perfect binary search tree.

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

      iv. (3 points) N get operations on a java.util.ArrayList of length N.

   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 to each other in the list and circle them.

3. (17 points total) Expression Trees

   a. (13 points) Given a mathematical expression in postfix notation (RPN) apply the expression tree generation algorithm. Show the state of the stack at each step as you process each token of the expression. Draw the final expression tree that is popped off at the end.

      4 5 7 * 6 4 - - +

b. (4 points) Give an invalid postfix expression to be evaluated by the algorithm in part a. Your example must cause the stack to underflow before it reaches the end of the expression. (Note: You are only giving us the expression, you don't have to run through the algorithm.)

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) Write a standalone **recursive** Java method, *int calcHeight(TreeNode root)* that, given a reference to the root node of a binary tree, returns the height of the tree. Assume a standard Binary TreeNode implementation with left child and right child references (there is no height field stored in these nodes). An empty tree will return a height of -1. (You may find the Math.max function useful in your implementation; it takes in two ints and returns the larger of the two.)

int calcHeight(TreeNode root) {

}

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 {

    public Node(int d, Node p, Node n) {
        data = d; prev = p; next = n;
    }

    Node prev;
    Node next;
    int data;
  }

  Node head;
  Node tail;

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

}
```

7. (11 points total) Binary Search Trees:

   a. (7 points) Starting with an empty Binary Search Tree, show the tree after inserting each of the following values sequentially: 7, 1, 4, 3, 6, 5, 2.

   b. (4 points) Take the tree resulting from part a and perform a full remove operation on the value 4. When removing, you must pick your replacement from the **left** subtree if there are two children. Draw the resulting tree.