CSE 373 A 18Sp: Midterm April 27 2018

Name_____

Student ID#_____

Instructions

- This test is closed book, closed calculators, closed electronics.
- You are permitted a single sheet of 8.5" x 11" notes
- Do not start the exam until told to do so. You will receive a 10 point deduction if you work before the exam starts or keep working after the instructor calls for papers
- Please write your answers neatly in the provided space. Be sure to leave some room in the margins as we will be scanning your answers and it's easy to miss markings too close to the edge of the paper.
- If you need extra space, feel free to use the back of the paper
- If you have a question, please raise your hand to ask the course staff for clarification

Rubric

#	Problem	Points	Score
1	AVL Tree Insertions	13	
2	Hash Table Insertions	17	
3	Asymptotic Analysis	20	
4	Modeling Code	15	
5	Testing and Debugging	20	

Advice

- Read questions carefully, you might be asked to refer to work you did earlier in a later part of a question.
- Write down assumptions, thoughts and intermediate steps so you can get partial credit. Clearly circle your final answer.
- The questions are not necessarily in order of difficulty, skip around.

1. AVL Tree Insertions

a. Insert the following sequence of values into an AVL tree **in the given order**, draw the resulting tree. **As you add nodes to the tree pay close attention to which nodes cause single and double rotations.** Please show the different stages of the tree if you wish to receive partial credit, you may use the back of this paper if necessary.

50, 40, 45, 47, 20, 46, 18, 25, 22, 21

b. Which of the above insertions caused a rotation? Indicate whether each of the rotations required a single or double rotation.

Single Rotation Insertions:

Double Rotation Insertions:

2. Hash Tables

a. Consider the following sequence of keys:

```
"hash", "tables", "are", "fun", "if", "you", "understand", "them"
```

Assume your table uses the following hash function:

```
public int getHashIndex(String key) {
   return key.length() % table.size;
}
```

Insert the above keys **in the given order** into a hash table with a capacity of 5. Use **separate chaining** to handle collisions. Do not worry about resizing.

0	1	2	3	4

b. What is the load factor of the table in part a after all the insertions? Based on this would it be appropriate to resize the array? **Briefly justify your answer.**

c. Consider the following sequence of keys:

```
22, 758, 103, 542, 707, 88, 57
```

Assume your table uses the following hash function:

```
public int getHashIndex(int key) {
   return key % table.size;
}
```

Insert the above keys **in the given order** into a hash table with a capacity of 10. Use **linear probing** to handle collisions. Do not worry about resizing.

0	1	2	3	4	5	6	7	8	9

d. What is the load factor of the table in part c after all the insertions? Based on this would it be appropriate to resize the array? **Briefly justify your answer.**

3. Asymptotic Analysis

a. Consider the following functions that represent the asymptotic growth rate of different algorithms as they are given larger and larger input sets. For each of the below, state the **simplified tight O bound** in terms of n.

Runtime	O(f(n))
$a(n) = 5\log_2(n)$	
$b(n) = 2n + 5n^2$	
$c(n) = 50n - 4$	
$d(n) = 8{,}000{,}000$	
$e(n) = 3^n + 4n$	

b. List the above functions in order from fastest growing (1) to slowest growing (5).

Relative Growth Rate	f(n)
1 (fastest growing)	
2	
3	
4	
5 (slowest growing)	

c. Based on the above, indicate whether the following statements are “true” or “false”

Statement	TRUE or FALSE
$a(n) \in O(b(n))$	
$e(n) \in O(b(n))$	
$c(n) \in \Omega(d(n))$	
$a(n) \in \Omega(e(n))$	
$d(n) \in \Theta(c(n))$	
$b(n) \in \Theta(b(n))$	

d. Find a **c** and $\mathbf{n_0}$ that show that **$c(n) \in O(b(n))$**. Show how you chose your c and n_0, no credit will be given for answers without work. Note: work should be algebraic, not just guess and check.

e. Imagine you are selecting from the above algorithms which to implement in a project on which you are working. Which of the five algorithms would you pick and why? Is this the best pick for all sizes of input? Briefly justify your answer in English (algebra unnecessary).

4. Modeling Code

Consider the following code. In the code there is a HashMap used to store data. You can assume the HashMap **contains no collisions**.

```
public int mystery(int n, HashMap<Integer, Integer> dictionary) {
   if (n <= 10) {
      return dictionary.get(n);
   } else if (n <= 50) {
      int result = 0;
      for (int i = 0; i < n; i+= 10) {
         for (int j = 0; j < 10; j++) {
             result++;
         }
      }
      return dictionary.get(n);
   } else {
      int result = 0;
      for (int i = 0; i < n * n; i++) {
         result++;
      }
      return result + mystery(n / 2);
   }
}
```

a.) What would be returned from the following calls to this method?

```
mystery(5);

mystery(30);

mystery(100);
```

b.) Provide a mathematical function T(n) that represents the **simplified, worst-case runtime** of the following code. Note: should your answer include a summation or a recurrence you do not need to find a closed form for your model.

5. Debugging Code

Consider the following method which accepts a single AVLNode and returns true if that node represents a valid AVL tree, false otherwise:

```
public boolean isAVL(AVLNode node) {
   if (node == null) {
      return true;
   } else if (node.left.height – node.right.height > 1) {
      return false;
   } else if (node.left.data > node.data ||
              node.right.data < node.data) {
      return false;
   } else {
      return isAVL(node.left) || isAVL(node.right);
   }
}
```

The following class is included in the same project:

```
public class AVLNode {
   public int data;
   public int height;
   public AVLNode left;
   public AVLNode right;

   …
}
```

a.) There are at least 4 fundamental bugs in this code which would cause it to not return the correct value. **Explain 2 of those situations** and how these bugs impact the whether the code correctly validates a given AVL Tree.

b.) For each of the 2 bugs you explained in part a, draw out a possible input tree that would illustrate the results you describe whether that is a thrown exception or an incorrect return value. This means you should **draw 2 different possible inputs**.

c.) For the 2 bugs and input trees you describe in parts a and b write out **pseudocode** unit tests that you would implement to test whether the code correctly handles these cases. You must write at least **1 test per bug you have described**. Your pseudocode should mimic the structure of real code but does not have to use real Java syntax. You can describe any functionality and operations you wish to use. Your pseudocode tests should include:

1. **Descriptive method names** that indicate what that test is evaluating
2. **An assertion** that compares the results of the code to the expected results

d.) There are 5 distinct categories of test cases we have discussed in class. Choose 3 of them and draw or describe 2 different inputs that could be tested to represent those test cases. This means **you need to describe at least 6 different inputs** to test. **THESE CANNOT BE THE SAME INPUTS YOU USED IN PART B.**

Extra Credit:

For 1 extra credit point write a pseudocode method that **in your ideal, magical world** you wish the course staff to use when deciding your grade.