CS 1331 Final Exam Practice

Fall 2014

Name (print clearly): _____

Signature: _____

GT account username (gtg, gth, msmith3, etc): _____

- Signing signifies you are aware of and in accordance with the **Academic Honor Code of Georgia Tech**.
- Calculators and cell phones are NOT allowed.
- This is an object-oriented programming test. Java is the required language. Java is case-sensitive. DO NOT WRITE IN ALL CAPS. A Java program in all caps will not compile. Good variable names and style are required. Comments are not required.

Question	Points per Page	Points Lost	Points Earned	Graded By
Page 1	8	-	=	
Page 2	8	-	=	
Page 3	8	-	=	
Page 4	10	-	=	
Page 5	10	-	=	
Page 6	0	-	=	
Page 7	0	-	=	
Page 8	0	-	=	
Page 9	0	-	=	
Page 10	0	-	=	
TOTAL	44	-	=	

1. **Multiple Choice** Circle the letter of the correct choice.

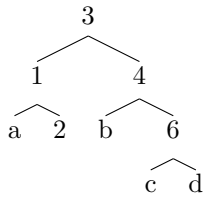
[2] (a) The time complexity of adding an element to the front of a singly-linked list with only a reference to the first node is ____.

A. $O(1)$
B. $O(\log n)$
C. $O(n)$
D. $O(n^2)$
E. $O(2^n)$

[2] (b) The time complexity of finding an element in a singly-linked list with only a reference to the first node is ____.

A. $O(1)$
B. $O(\log n)$
C. $O(n)$
D. $O(n^2)$
E. $O(2^n)$

Given the following tree that conforms to the binary search tree property:



[2] (c) At which position would the element 5 be inserted?

A. a
B. b
C. c
D. d

[2] (d) At which position would the element 7 be inserted?

A. a
B. b
C. c
D. d

 Points available: 8 - points lost: _____ = points earned: _____. Graded by: _____

2. **Multiple Choice** Circle the letter of the correct choice.

[2] (a) What is true about this code?

```
public static int fac(int n) {
    return n * fac(n - 1);
}
// ...
int fac5 = fac(5);
```

A. Compiles and runs without errors or exceptions.

B. Compiles but program terminates with an error or exception.

[2] (b) What is true about this code?

```
public static int fac(int n) {
    if (n <= 1) return 1;
    else return n * fac(n + 1);
}
// ...
    int fac5 = fac(5);
```

A. Compiles and runs without errors or exceptions.

B. Compiles but program terminates with an error or exception.

[2] (c) Given the following recursive method:

```
private static int bs(int[] array, int queryValue, int lo, int hi) {
    if (lo > hi) return -1;
    int middle = (lo + hi)/2;
    if (queryValue == array[middle]) return middle;
    else if (queryValue > array[middle]) {
        return bs(array, queryValue, middle + 1, hi);
    } else {
        return bs(array, queryValue, lo, middle - 1);
    }
}
```

and an array defined as `int[] a = {4,7,5,2,3,9,6,1,8}`, what would be returned by the call `bs(a, 5, 0, a.length - 1)`?

A. -1

B. 2

C. 3

[2] (d) What is the worst-case time complexity (Big-O) of the following method?

```
public int f(int n) {
    int s = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            s += j;
      }
    }
    return s;
}
```

A. $O(1)$

B. $O(\lg n)$

C. $O(n)$

D. $O(n^2)$

3. **Multiple Choice** Circle the letter of the correct choice.

```
public static int f(int n) {
    if (n < 0) throw new IllegalArgumentException("n < 0");
    if (n <= 1) {
        return n;
    } else {
        return f(n - 1) + f(n - 2);
    }
}
```

[2] (a) Given the method `f` above, what is `f(5)`?

A. 0
B. 4
C. 5
D. 120

[2] (b) Given the method `f` above, what is `f(4)`?

A. 0
B. 3
C. 4
D. 24

```
1: public class ArrayListStack<E> {
2:     private ArrayList<E> elems = new ArrayList<>();
3:
4:     public void push(E item) {
5:
6:     }
7:
8:     public E pop() {
9:
10:    }
11:
12:    public boolean isEmpty() {
13:        return elems.isEmpty();
14:    }
15:}
```

[2] (c) Given the partial `ArrayListStack` implementation above, which of the following statements for line 5 would implement `push` in $O(1)$ time? Do not consider any particular implementation for `pop`.

A. `elems.add(item);`
B. `elems.add(0, item);`
C. `return elems.remove(elems.size() - 1);`

[2] (d) Given the partial `ArrayListStack` implementation above, which of the following statements for line 5 would implement `push` in $O(n)$ time? Do not consider any particular implementation for `pop`.

A. `elems.add(item);`
B. `elems.add(0, item);`
C. `return elems.remove(elems.size() - 1);`

 Points available: 8 - points lost: _____ = points earned: _____. Graded by: _____

[10] 4. Assume you have a `LinkedList` class defined as follows:

```
public class LinkedList<E> {

    // Add a new item to the back of this list.
    public void addBack(E item) { ... }

    // Return the item at the front of this list and remove it from the list.
    public E removeFront() { ... }

    // Return true if this list has no elements, false otherwise.
    public boolean isEmpty() { ... }
}
```

Complete the following definition of a `Queue` class, which uses the `LinkedList` class defined above. You may use only the space provided between the curly braces for each method, which is all you need.

```
public class Queue<T> {
    private LinkedList<T> list;

    public Queue() {



    }

    public void enqueue(T item) {



    }

    public T dequeue() {



    }

    public boolean isEmpty() {



    }
}
```

[10] 5. Complete the following definition of a `BinarySearchTree` class by providing the code for the helper method `buildString(Node<E> node, StringBuilder accum)`. `buildString` is a helper method for `toString`, which prints the elements of a binary tree in order like this: `[1,2,3,4,5,]`.
To append a `String` value to a `StringBuilder` instance, use `StringBuilder`'s `append` method, as in `stringBuilder.append("new text")`. You may use only the space provided between the method's curly braces, which is all you need.

```
public class BinarySearchTree<E extends Comparable<E>> {
    private class Node<E> {
        E item;
        Node<E> left, right;

        Node(E item, Node<E> left, Node<E> right) {
            this.item = item;
            this.left = left;
            this.right = right;
        }
    }
    private Node<E> root;

    public void add(E item) { root = insert(item, root); }

    private Node<E> insert(E item, Node<E> node) {
        if (node == null) { return new Node<E>(item, null, null); }
        else if (item.compareTo(node.item) < 0) {
            node.left = insert(item, node.left);
            return node;
        } else {
            node.right = insert(item, node.right);
            return node;
        }
    }
    public String toString() {
        return "[" + buildString(root, new StringBuilder()).toString() + "]";
    }
    // Complete this method
    private StringBuilder buildString(Node<E> node, StringBuilder accum) {









    }
}
```

 Points available: 10 - points lost: _____ = points earned: _____. Graded by: _____