Computer Science 141 – Final Exam

April 29, 2004

General Directions. This is an open-book, open-notes, open-computer test. However, you may not communicate with any person, except me, during the test. You have the full exam period (3 hours) in which to do the test. Put your answer to each question in the space provided (use the backs of pages if you need more space). Be sure to **show your work**! I give partial credit for incorrect answers if you show correct steps leading up to them; conversely, I do not give full credit even for correct answers if it is not clear that you understand where those answers come from. Good luck.

This test contains 7 questions on 7 pages.

Question 1 (10 Points). The LineDrawing class that we have used in some labs has only one constructor, which takes no parameters, and which initializes the pen's position, heading, and color to default values. Imagine a "BetterLineDrawing" subclass of LineDrawing, which provides a constructor that allows clients to specify exactly where the pen should start, what direction it should be heading in, and what color it should be drawing in. This constructor takes the pen's initial position (two real numbers, representing the *x* and *y* coordinates), heading (a real number), and color (an object from class java.awt.Color) as parameters. Write this constructor (pseudocode is OK).

Question 2. Imagine ordered binary trees that contain months of the year. Months are ordered in the usual calendar order, i.e., January comes before February, February comes before March, etc.

Part A (10 Points). Draw an ordered binary tree of months that contains April, January, December, October, February, June, and August.

Part B (10 Points). What is the minimum height that a tree of 7 months could have?

Part C (5 Points). What is the maximum height that a tree of 7 months could have?

Question 3 (20 Points). Consider ordered lists, i.e., lists whose elements have a "less than" relation. Ordered list *A* is "lexicographically less than" ordered list *B* if (but only if) any of the following are true:

1. List *A* is empty but *B* is not
2. Lists *A* and *B* are both non-empty, and the first item of *A* is less than the first item of *B*
3. Lists *A* and *B* are both non-empty, their first items are equal and the tail of *A* is lexicographically less than the tail of *B*.

For example, the empty list is lexicographically less than (1 2 3); (1 2 3) is lexicographically less than (2 3); (1 2 3) is lexicographically less than (1 4 5).

Write (pseudocode is OK) a recursive "lessThan" method for a subclass of List that takes a list, *B*, as its parameter, and returns true if the list executing the method is lexicographically less than *B*, and returns false otherwise.

Question 4 (15 Points). Here (in a combination of Java and pseudocode) is an algorithm that supposedly determines whether or not all the objects in a binary tree are strings. If so, the algorithm should return true, if not it should return false.

```
// In some subclass of OrderedTree...
public boolean allStrings() {
   if ( this.isEmpty() ) {
      return true;
   }
   else if ( this.getRoot() is not a string ) {
      return false;
   }
   else {
      return       this.getLeft().allStrings()
             &&    this.getRight().allStrings();
   }
}
```

Prove that this algorithm really does return true if all objects in a tree are strings, and false otherwise.

Question 5 (15 Points). Susan Science wants to know how long it takes one of our List objects to handle an “addItem” message, so she writes the following code:

```
List testList = new List();
long start = System.currentTimeMillis();
for ( int i = 0; i < 1000; i++ ) {
   testList.addItem( “foo” );
}
long end = System.currentTimeMillis();
long controlStart = System.currentTimeMillis();
for ( int i = 0; i < 1000; i++ ) { }
long controlEnd = System.currentTimeMillis();
System.out.println( “start = “ + start );
System.out.println( “end = “ + end );
System.out.println( “control start = “ + controlStart );
System.out.println( “control end = “ + controlEnd );
```

When Susan runs this program, it prints

```
start = 1905
end = 1916
control start = 1916
control end = 1917
```

How long do you estimate that it takes the list to handle one “addItem” message? Include the units in which time is measured, and remember to show your work.

Question 6 (20 Points). Suppose I want a subclass of OrderedTree that allows me to initialize trees of a client-specified height, and filled with dummy data. I write the following constructor for my subclass:

```
public MySubclass( int height ) {
   super();
   if ( height > 0 ) {
      this.setRoot( “Dummy Data” );
      this.setLeft( new MySubclass( height – 1 ) );
      this.setRight( new MySubclass( height – 1 ) );
   }
}
```

Derive the asymptotic execution time of this algorithm, as a function of the height of the tree. You may assume that the superclass's constructor, and the “setRoot”, “setLeft”, and “setRight” methods, all run in constant time.

Question 7 (15 Points). A program is measuring the execution time of some operation, and reports the following measurements:

```
17 mS
2 mS
2 mS
5 mS
1 mS
2 mS
```

Classify each of the above measurements as an outlier or not an outlier, and say why you classify each measurement as you do. Then give the number that you would use as “the” execution time for this operation. Explain briefy how you arrived at this number.