Computer Science 141 — Exam 1

October 7, 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 class period (75 minutes) 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 4 questions (one with two parts) on 6 pages.

Question 1 (15 Points). Write (pseudocode is fine) a recursive algorithm for the text's robots that takes a heading as its parameter, and makes a robot turn left until it is pointing in that direction, then turn back to the right until it is pointing in its original direction. For example, if the robot was initially facing south, and the parameter was north, the robot would go through the following sequence of headings:

South (initial heading)
East (after first left turn)
North (after second left turn)
East (starting to turn back to the right)
South (back to original heading)

Assume your algorithm is a method of a subclass of `Robot`. If you want a concrete way to declare the parameter to this method, recall that the robot uses `int`s to represent headings.

Question 2 (15 Points). Consider the following piece of code:

```
class SuperRobot extends Robot {

   public void turnAround() {
      this.turnLeft();
      this.turnLeft();
   }

   public void paintAndMove( java.awt.Color c ) {
      this.paint( c );
      this.move();
   }
}
```

Write a paragraph about what this code does or defines. Use the following terms (or near variants, e.g., plurals, etc.) in your paragraph, in ways that are technically correct for object oriented programming:

Subclass

Inherit

Message

Method

(This question tests how well you understand the terminology of object oriented programming; your answer will be graded mainly on correct use of each of the above terms, and hardly at all on other insight into the code.)

Question 3. Recall the `SuperArray` class, which provided a number of examples of recursion and induction in lectures. Here is an algorithm that adds up the values in positions 0 through i of a `SuperArray`:

```
// In class SuperArray...
int addValues( int i ) {
   if ( i == 0 ) {
      return A[0]
   }
   else {
      return this.addValues( i-1 ) + A[i]
   }
}
```

Part A (15 Points). Prove that the message `addValues(i)` sent to a `SuperArray` returns A[0] + A[1] + ... + A[i], where A is the `SuperArray`'s internal array. Assume as a precondition that $i \geq 0$.

Part B (15 Points). Derive an expression for the number of times the algorithm accesses an element of A (i.e., the number of "A[...]" expressions the algorithm evaluates), as a function of i. You may assume that $i \geq 0$. Be sure to show all the work leading up to the expression, including any proofs.

Here is the algorithm again, for reference:

```
// In class SuperArray...
int addValues( int i ) {
   if ( i == 0 ) {
      return A[0]
   }
   else {
      return this.addValues( i-1 ) + A[i]
   }
}
```

Question 4 (15 Points). Suppose that in working with our `Robot` class, I have noticed that robots seem to take about half a second to handle a `move` message. Curious about whether this is really the case, I decide to do a formal experiment. So I write the following code:

```
final int n = ....;
RobotRoom room = new RobotRoom( “25 25” );
Robot mover = new Robot( 1, 23, Robot.NORTH, room );
long start = System.currentTimeMillis();
for ( int i = 1; i <= n; i++ ) {
   mover.move();
}
long end = System.currentTimeMillis();
System.out.println( end – start );
```

Running this program several times for each of several values of n yields the following data:

n	***Times (mS)***
5	2501, 2500, 2502, 2501
10	5010, 5000, 5010, 5020
15	7520, 7520, 7530, 7530
20	10000, 10020, 10010, 10010

Do these data support or refute the hypothesis that a `move` message takes 500 mS to execute? Explain your answer, including showing data analysis to justify it.