Computer Science 141 -- Exam 1

March 2, 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 5 questions on 5 pages.

Question 1 (15 Points). The following algorithm supposedly returns 10^n, where *n* is the algorithm's parameter. Prove that it really does so for every natural number *n*.

```
static int power( int n ) {
   if ( n == 0 ) {
      return 1;
   }
   else {
      return 10 * power( n – 1 );
   }
}
```

Question 2 (20 Points). Write (pseudocode is fine) an algorithm that takes a natural number, *n*, as its only parameter, and that behaves as follows:

- If *n* is even, the algorithm prints the even numbers from *n* down to 0, in descending order, followed by the odd numbers from 1 up to *n*–1, in ascending order.
- If *n* is odd, the algorithm prints the odd numbers from *n* down to 1, in descending order, followed by the even numbers from 0 up to *n*–1, in ascending order.

For example, if invoked with parameter 6, the algorithm would print

```
6 4 2 0 1 3 5
```

If invoked with parameter 5, the algorithm would print

```
5 3 1 0 2 4
```

Question 3 (15 Points). Here is an algorithm that takes a string as its parameter, and returns a new string that contains basically the same text, except that all lowercase letters in the original have been turned to uppercase:

```
static String capitalize( String s ) {
   if ( s.length() == 0 ) {
      return s;
   }
   else {
      return    s.charAt(0).toUpperCase()
             + capitalize( s.substring(1,s.length()) );
   }
}
```

Derive an expression for the number of `String` messages (`length`, `charAt`, and `substring`) this algorithm sends, as a function of the length of `s`. Be sure to show all the steps in the derivation, including any proofs.

Question 4 (10 Points). Phineas Phoole has written the following extension to our Robot class. As you will see when you read his comments, Phineas isn't very good at using the technical terminology of object oriented programming. Rewrite the comments to correct any errors in use of technical terms related to object oriented programming:

```
class BetterRobot extends Robot {

   // This is a class of “Robot”. It contains subclasses
   // “spin” and “doubleStep”. Sending this class a “spin”
   // method makes a robot turn left n times, where n is the
   // method's object. Sending the class a “doubleStep”
   // method makes the robot move forward twice.

   public spin( int n ) {
      for ( int i = 0; i < n; i++ ) {
         this.turnLeft();
      }
   }

   public void doubleStep() {
      this.move();
      this.move();
   }
}
```

Question 5 (15 Points). Suppose you want to test the hypothesis that the time it takes Java's `println` method to print a string is proportional to the number of characters in the string. So you write the following main method:

```
public static void main( String[] args ) {
  String test = ...      // Strings of various lengths
  long start = System.currentTimeMillis();
  System.out.println( test );
  long end = System.currentTimeMillis();
  System.out.println( end – start );
}
```

You run this program with string `test` having various lengths, and collect the following raw data (in milliseconds):

For `test` containing 10 characters: 3, 2, 3, 3, 2, 2

For `test` containing 20 characters: 4, 5, 7, 5, 5, 4

For `test` containing 40 characters: 10, 10, 9, 10, 11, 10

Analyze these data to determine whether they are consistent with the hypothesis that the time to print a string is proportional to the string's length. Briefly explain your analysis, and say whether it supports the hypothesis or not.