Computer Science 141 -- Exam 2

November 19, 2003

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 (50 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 on 3 pages.

Question 1 (15 Points). Write (pseudocode is fine) a recursive algorithm that tests whether all the items in a list are equal to each other. Your algorithm should be a method of an "ExtendedList" subclass of "List". It should return True if all the items in the list are equal, and False if not. For example, if run on the list of strings [“dog” [“dog” [“dog” []]]], your algorithm would return True, while if run on the list [“dog” [“cat” [“dog” []]]], it would return False.

Question 2 (15 Points). Imagine that you have invented an algorithm for sorting lists. Believing that your algorithm sorts a list of n items in Theta(n^2) time, you code it and measure its execution time on lists of various sizes. Here are the list sizes and average sorting times you collect:

List Size	***Average Sorting Time (mS)***
1	4
2	8
5	20
10	30
20	80
50	125
100	200

Do these data support your Theta(n^2) hypothesis? Why or why not?

Question 3 (15 Points). Suppose you have a list whose items are numbers. The following algorithm supposedly computes the sum of all the numbers in such a list (the sum of all the numbers in an empty list is defined to be 0):

```
// In class NumberList
sum()
  if this.isEmpty()
    return 0
  else
    return this.first() + this.rest().sum()
```

Prove that this algorithm really does return the sum of the items in a list.

Question 4 (15 Points). Here is an algorithm that counts the number of occurrences of object *x* in a list:

```
// In class SearchableList...
int count( Object x )
  if ( this.isEmpty() )
    return 0
  else if ( this.first().equals(x) )
    return 1 + this.rest().count( x )
  else
    return this.rest().count( x )
```

Derive (and prove, if necessary) a formula for the number of primitive List messages ("isEmpty," "first," and "rest") that this algorithm sends, in terms of the length of the list.