Computer Science 141 — Exam 2

November 9, 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 (10 Points). Here is code that uses a stack, `s`, to swap objects `x` and `y` (i.e., to make `x` equal to `y`'s original value, and vice versa):

```
s.push(x)
s.push(y)
x = s.pop()
y = s.pop()
```

Does it matter whether `s` is empty before executing this code in order for it to really swap `x` and `y`? Why or why not?

Question 2 (15 Points). Here is pseudocode for a method in a “list of numbers” subclass of our `List` class. The method takes a second list of numbers as its parameter, and returns the sum of the products of corresponding numbers from `this` and the parameter. The method has a precondition that both lists have the same number of elements. For example, if `nums1` were the list (2, 4, 6) and `nums2` were the list (10, 20, 30), then `nums1.sumOfProducts(nums2)` would return 2•10 + 4•20 + 6•30 = 280.

Prove that this algorithm really does compute the sum of the products of the numbers in two lists. Recall that the sum of no numbers is 0.

```
// In ListOfNumbers, a subclass of List:
double sumOfProducts( ListOfNumbers other )
   if ( this.isEmpty() )
      return 0.0
   else
      product = this.getFirst() * other.getFirst()
      return product + this.getRest().sumOfProducts( other.getRest() )
```

Question 3 (20 Points). Phineas Phoole has designed an algorithm to capitalize all the strings in a list of strings. His algorithm consists of two methods, and looks like this, in pseudocode (assuming a `capitalize` message to strings that capitalizes them):

```
// In some subclass of List:
void capitalizeAllItems()
   if ( this.length() > 0 )
      this.getFirst().capitalize()
      this.getRest().capitalizeAllItems()

int length()
   if ( this.isEmpty() )
      return 0
   else
      return 1 + this.getRest().length()
```

Derive (and prove, if necessary), an expression for the number of primitive List messages (getFirst, getRest, isEmpty) that this algorithm sends, in terms of the list's length.

Question 4 (15 Points). Imagine you have just invented an algorithm to sort lists, and you believe that your algorithm sorts an n-item list in time $\Theta(n^2)$. To test this hypothesis, you measure how long the algorithm takes to sort lists of various sizes, gathering the following data:

List Size	***Average Sorting Time (mS)***
1	10
5	100
10	200
50	2000
100	5000
500	25000

Are these measurements consistent with your $\Theta(n^2)$ hypothesis? Why or why not?

Question 5 (15 Points). Imagine a subclass of our `List` class that represents lists of words. Every item in such a list is a string. Write a recursive method `checkSpelling` that checks the spelling of each word in such a list by looking each word up in a “dictionary.” The dictionary is a parameter to `checkSpelling`, and is simply another list of words. The message `words.checkSpelling(dict)` returns True if every word in `words` is also in `dict`, and False otherwise. Here is the header for `checkSpelling`; fill in the rest of the method (pseudocode is fine):

```
// In ListOfWords, a subclass of List:
boolean checkSpelling( ListOfWords dictionary )
```