

CS 101 Computer Science I (Spring 2001)
Lab 3: February 7-9, 2001

1. Write a recursive procedure called "**double-all**" that takes a list of numbers and returns a list in which each of the original list items has been doubled. Examples:

```
(double-all '())           ==> ()
(double-all '(3))         ==> (6)
(double-all '(3 4.2 7))   ==> (6 8.4 14)
```

2. Write a recursive function called "**get-nth**" that takes a positive integer **n** and a list **lst** as arguments and returns the **n**th element of **lst**. If the length of **lst** is less than **n**, the function returns the symbol **fail**.

```
(get-nth 1 '(a b c))      ==> a
(get-nth 2 '(a b c))      ==> b
(get-nth 3 '(a b c))      ==> c
(get-nth 4 '(a b c))      ==> fail
```

3. Write a recursive predicate called "**prefix?**" that takes two arguments, both of which are lists of symbols or numbers. The predicate returns **#t** if the first list is a prefix of the second, and returns **#f** otherwise.

```
(prefix? '() '(foo bar))  ==> #t
(prefix? '(7) '(7 10))    ==> #t
(prefix? '(7) '(10 7))    ==> #f
(prefix? '(a b c) '(a b c d e)) ==> #t
(prefix? '(a b c) '(a b x y z)) ==> #f
```

4. Suppose we want to tell as quickly as possible whether one list is the same length as another. Of course we could use the length function to compute the length of each list and then test whether the lengths are equal. What if one list has five elements and the other has five million elements? We would be wasting lots of time computing the length of the second list. Write a recursive function called "**compare-lengths**" that takes two lists as arguments and returns a symbol that indicates whether the first list or second list is shorter, or if they are the same length:

```
(compare-lengths '(a b c) '(a b c d e)) ==> first-is-shorter
(compare-lengths '(a b c d e) '(a b c)) ==> second-is-shorter
(compare-lengths '(a b c d) '(b c d e)) ==> equal-lengths
```