

COMP 210, Spring 2000, Homework 2
Due Friday, February 18, 2000 at the start of class

Before you start the homework, you should remind yourself of our General Advice, Advice on Homeworks, and Grading Guidelines. All are available from the class web site (<http://www.owl.net.rice.edu/~comp210>).

For this assignment, you should follow all the steps of the design methodology and include the results of each step as comments or code in the final materials that you submit. (For example, write your template as a comment—at the appropriate point in the development sequence—and copy it over when you fill it in.)

1. (2 pts) Consider the domain of natural numbers, as defined in Lecture 4 (notes are on-line). Write a program **multiply** that takes two natural numbers and returns their product. Your program may not use the built-in multiply function; instead, you should use addition and subtraction to compute the answer.

Show all the steps in the design methodology. Hand evaluate two cases. Use DrScheme to evaluate them, as well as other test cases.

2. (3 pts) Given the definition for list-of-sym-and-num from Lecture 10 (notes are on-line), and the mapping of a recipe onto a list-of-sym-and-num, write a program **substitute** that takes three arguments—a list-of-sym-and-num that represents a recipe, a symbol **old** and a symbol **new**. The program **substitute** should create a new recipe in which all occurrences of **old** are replaced with occurrences of **new**.

Show all the steps in the design methodology. Hand evaluate two cases. Use DrScheme to evaluate them, as well as other test cases.

3. (5 pts) We can define a **list-of-list-or-symbol** as

```
;; a list-of-list-or-symbol is one of
;;   – empty, or
;;   – (cons f r)
;;     where f and r are both list-of-list-or-symbol
;;   – (cons f r)
;;     where f is a symbol and r is a list-of-list-or-symbol
```

```
;; example data
  (cons 'fee
        (cons
          (cons 'fie
                (cons 'foe empty))
          (cons 'fum empty)
        ))
```

- a) Write a program **symbol-count** that takes a list-of-list-or-symbol and returns the number of symbols occurring in the input list. For the example data given earlier, **symbol-count** would produce 4.
- b) Write a program **list-count** that takes a list-of-list-or-symbol and returns the number of occurrences of **empty** in the input list. For the example data given earlier, the program would produce 2.
- c) Write a program **flatten** that consumes a list-of-list-or-symbol and produces a new list-of-symbol that has all the symbols from the list-of-list-or-symbol, in their order of appearance. For the example data given earlier, the program would produce

```
(cons 'fee
      (cons 'fie
            (cons 'foe
                  (cons 'fum empty))))
```