COMP 210, Spring 2001, Homework 3
Due Wednesday, February 7, 2001 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.owlnet.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. (4 pts) **Programs on Lists**
   Write down a data definition for lists of numbers. Now, develop the following programs that consume a list of numbers:
   a) A function **`summation`** that returns the sum of all the numbers in the list
   b) A function **`all-positive?`** that returns **`true`** if and only if every number in the list is greater than or equal to zero.
   c) A function **`count-positives`** that returns the number of positive numbers in the list.
   d) A function **`mostly-positive`** that returns true if the list contains more positive numbers (≥ 0) than negative numbers (< 0).

2. (3 pts) **Digital Telephone Directory**
   Once a year, the local telephone company publishes a directory. For our purposes, the directory is a list of pairs. The pair consists of a symbol, called the key, and a phone number, represented as a number.
   a) Write out the data definitions for this simple online phone directory. You should have one data definition for pairs and a second data definition for directories.
   b) Write a program **`lookup`** that consumes a symbol and a phone directory and produces a phone number. The program **`lookup`** should examine the list for a key that matches the symbol given as input. If it finds a key that matches the input symbol, it returns that phone number. If no matching record is found, it returns zero.

   Be sure to create a reasonable set of test data and enter it in the definitions window of DrScheme.

   Schemers sometimes call a directory in this form an association list.

3. (3 pts) **Programs that return Lists**
   a) A function **`positive-elements`** that consumes a list of numbers and returns a list containing those elements of the list that are positive.
   b) A function **`even-positions`** that consumes a list and returns those elements in the even-numbered positions in the list assuming that the first element has position 1, the second element has position 2, *etc*. Hence,

```
(even-positions (cons 1 (cons 2 (cons 3 (cons 4 empty))))) =
(cons 2 (cons 4 empty))
```

4. Extra Credit (3 pts) **Programs that involve more complex templates**

a) A function **`merge`** that consumes two lists **`l1`** and **`l2`** of numbers in ascending order and returns a list containing the elements of both **`l1`** and **`l2`** in ascending order. Hence,

```
(merge  (cons 1 (cons 6 (cons 12 empty)))
        (cons –1 (cons 4 (cons 4 empty))))
= (cons –1 (cons 1 (cons 4 (cons 4 (cons 6 (cons 12 empty))))))
```

b) A function **`split`** that takes a list **`l`** and returns a pair of lists **`u`** and **`v`** of nearly the same length (differing by at most one) which together contain exactly the same elements as **`l`**. Hence,

```
(split (cons 17 (cons 2 (cons –1 (cons 6 (cons –5 empty))))))) =
(make-Pair (cons –5 (cons –1 (cons 17 empty))) (cons 2 (cons 6 empty)))
```

is a possible result for **`split`**.

c) A function **`sort`** that sorts a list into ascending order using **`split`** and **`merge`** as help functions. In **`sort`**, the number of **`cons`** operations performed in sorting a list of *n* elements must be proportional to *n * log(n)* not *n*n.*