

# Homework 4

*Due: 20 Jan 2009*

For all problems in this assignment, use `map` or `filter` or `foldl` (aka `reduce`) as appropriate.<sup>1</sup> You're also encouraged to write helper functions wherever this makes the code more readable or more testable (usually both).

You won't necessarily be able to use `map` et al in every case, so don't bang your head too much if they don't seem to fit. Some of the problems are just about managing lists.

Include contracts for all the functions you write.

## Problem 4.1

Write the following functions, which operate on lists of various sorts:

- a. `only-alphabetic` makes a list containing those members of a given list that are alphabetic characters.
- b. `log-scale` makes a list containing the natural logarithm of all the values in a given list of numbers.
- c. `extract-numbers` constructs a list containing the numeric elements of a given list.

## Problem 4.2

- a. Write `(unique? lst)`, which determines whether all the values in `lst` are unique (i.e. that there aren't any duplicate values—use `equal?` to test this).
- b. Then write `(unique-values lst)`, which makes a list of all the unique values in `lst`.

Note that I am not expecting optimal big-O efficiency here.

---

<sup>1</sup>I got it backwards in class when we defined `mymap` and the others: in all three, the list argument comes *last* in the builtin definition. Sorry about that.

**Problem 4.3** ( $\times 2$ )

Write the following functions:

- a. `initials-list` makes a list of chars that are the initial letters from a given list of strings.
- b. `only-<5` makes a list containing those members of a given list that are less than five.
- c. `add-2-all` makes a list whose members are 2 greater than the elements of the given list.
- d. `exclude-alphabetic` makes a list containing those members of a given list that are *not* alphabetic characters.
- e. `subtract-4-all` makes a list whose members are 4 less than the elements of the given list.

**Problem 4.4**

A `point` is a structure containing two numbers named `x` and `y`, represented as a two-element list. Conveniently, in addition to `first`, DrScheme also defines for us `second`; `(second lst)` is precisely equivalent to `(first (rest lst))` but a lot easier to read.

Write the following functions, which operate on lists of `points`.

- a. `(first-quadrant lst)` makes a list with all the elements of `lst` that lie in the first quadrant (inclusive of axes).
- b. `(shift-all-right-one lst)` makes a list with all the elements of `lst` shifted rightwards by one unit (i.e. with an `x` value greater by 1).
- c. `(shortest-distance-to-origin lst)` calculates the distance between the origin and the closest point in the given list. (Distance is measured by the standard (Euclidean) distance formula.)