CS 457/557 Homework 3 – due 2pm, Tuesday, Oct. 18, 2005

Hand in your solutions on paper *and* email them to `cs457acc@cs.pdx.edu`. All the programs should be placed in a single `.hs` file, which can be the body of your email message or an attachment. It is *not* necessary to show evidence that you have loaded and tested your programs, but this is of course the only sensible way to make sure that you have found correct answers!

1. Write definitions for the `map` and `filter` functions in terms of list comprehensions. (Call them `map'` and `filter'` to avoid confusion with the existing Prelude functions.)

2.a. Suppose

```
h f g xs = map f (map g xs)
```

Give an alternative definition for `h` that builds no intermediate list.

b. Suppose

```
r p q xs = filter p (filter q xs)
```

Give an alternative definition for `r` that builds no intermediate list.

3. Suppose

```
map2 f = map (map f)
```

Use `map2` to define a function

```
upall :: [String] -> [String]
```

that converts every String in its argument list to upper-case. (Hint: Check the Prelude for useful functions on characters.)

4. Write implementations of the following functions making use of the `foldr` function instead of explicit recursion. (You can't use list comprehensions either.) I've specified variant forms of the usual names to avoid conflicts with the existing Prelude functions.

(a) `map' :: (a -> b) -> [a] -> [b]`

(b) `filter' :: (a -> Bool) -> [a] -> [a]`

(c) `(+++) :: [a] -> [a] -> [a]` (meant to act like `(++)`).

(d) `average :: [Int] -> Float` (using only *one* call to `foldr`).

5. Give a simple argument showing that it is impossible to express `foldr` in the form

```
foldr f i = map g
   where g = ...
```

6. Do Hudak problem 5.9. Try to write your solution as a `foldl` (although in fact a direct recursion may be more readable in this case).