Laboratory: Haskell (1)

Summary: We begin our exploration of the Haskell programming language.

Prerequisites: Tate, Sections 8.1 and 8.2.

Contents:

- Preparation
- Exercises
  - Exercise 1: Fibonacci
  - Exercise 2: Reversing Lists
  - Exercise 3: List Utilities
  - Exercise 4: Fibonacci, Revisited
  - Exercise 5: Polyexpressionism
  - Exercise 6: Reversing Lists
  - Exercise 7: Pairing Colors
- For Those with Extra Time

Preparation

a. Create a directory for the lab.

b. Open a browser window on Tate’s examples, in case you want to try any of them.

Exercises

Exercise 1: Fibonacci

Explore each of Tate’s Fibonacci examples and verify that they work the way you expect.

Exercise 2: Reversing Lists

This exercise is taken from Tate.

Write a function that computes the reverse of a list.

Exercise 3: List Utilities

a. Write mytake n lst, which extracts the first n elements of a list.
b. Write `mydrop n lst`, which drops the first `n` elements of a list.

c. Write `mynth n lst`, which computes the `n`th element of a list.

**Exercise 4: Fibonacci, Revisited**

At some point, you may have noted that it’s a bit strange that `fib` is defined as a function, given that we normally talk about the Fibonacci sequence. Write a recursive definition of the sequence.

In Lazy Scheme, we might write this as

```scheme
(define fib
  (cons 1
        (cons 1
              (map + fib (cdr fb))))))
```

You may find the following definition helpful.

```haskell
module Main where
  map2 f [] [] = []
  map2 f (x:xs) (y:ys) = (f x y):(map2 f xs ys)
```

**Exercise 5: Polyexpressionism**

*This exercise is taken from Tate.*

Write `allEven` in as many ways as you can.

**Exercise 6: Reversing Lists**

*This exercise is taken from Tate.*

Write a list comprehension to build a childhood multiplication table. The table would be a list of three-tuples where the first two are integers from 1-12 and the third is the product of the first two.

**Exercise 7: Pairing Colors**

*This exercise is taken from Tate.*

Write a function that builds two-tuples with all possible combinations of two of the colors black, white, blue, yellow, and red. Note that you should only include one of `(black, blue)` and `(blue, black)

**For Those with Extra Time**

If you find yourself with extra time, begin reading the next section of Tate.