Preliminaries

CS 340: Programming Paradigms and Patterns
Michael Saelee <lee@iit.edu>
Michael (Sae) Lee

- lee@iit.edu
- http://moss.cs.iit.edu
- Office: SB 226C
Agenda

- What is “PPP”?  
- Why Haskell?  
- Why take CS 340?  
- Administrivia  
  - Websites, References, Grading, etc.
What is PPP?

- Programming Paradigms and Patterns
Paradigm

- Model for how a program in a given language is organized, expressed, and/or executed

- e.g., procedural, imperative, object-oriented, functional, declarative

- We will be focusing on the **functional** paradigm
Pattern

- A reusable template for solving a common class of problem(s)
- May be paradigm/language specific, and typically as abstract as possible to encourage reuse
- We will focus on patterns used in functional programming:
  - e.g., generalized folds, functors, monadic parsing
Why Functional?

- Very different set of operating assumptions from your (likely) first model, imperative programming

- No state mutation $\rightarrow$ referential transparency

- Arguably easier to reason about (rigorously) and use for concurrency

- You’ll read a paper on this for Friday!
Haskell

- Our functional language of choice: **Haskell**
- **Pure**: purely functional — no state mutations
- **Statically typed**: types are checked at compile time
- **Lazy**: expressions aren’t evaluated until absolutely necessary
- Likely very different from another language you’ve used!
Why Haskell?

- It’s fun, surprising, and powerful!
- Learning a (different) new language gives you an entirely new way to think about and tackle problems
- Valuable, even if you don’t actually code the solution up in said language
A Taste of Haskell

```haskell
fibs = 0 : 1 : zipWith (+) fibs (tail fibs)

primes = filterPrime [2..]
    where filterPrime (p:xs) =
            p : filterPrime [x | x <- xs, x `mod` p /= 0]

quicksort :: Ord a => [a] -> [a]
quicksort []     = []
quicksort (p:xs) = (quicksort lesser) ++ [p] ++ (quicksort greater)
    where
        lesser = filter (< p) xs
        greater = filter (>= p) xs
```
Why take CS 340?

- You love to program
- You love programming languages
- You are frustrated with languages you currently know
- You want to learn new ways to reason about programming
  - This will help in later classes and your career
Topics (not exhaustive)

- Functional programming
- Haskell basics: syntax/semantics/libs
- Higher Order Functions
- Functors and other Abstractions
- Property-Based Testing
- Functional Reactive Programming
- Concurrency
§ Administrivia
Prerequisites

- I assume you are ...
  - fluent in some programming language
  - familiar with procedural & OO paradigms
- comfortable with development processes:
  - compilation, debugging, testing
Online resources

1. Course website: moss.cs.iit.edu/cs340
   - static information
   - lecture calendar, lab writeups, slides, screencasts, links, etc.
Online resources

2. Blackboard
   - only for grade reporting!
Online resources

3. Piazza: discussion forum
   - all class-related questions
   - monitored by TA
   - scales *way* better than e-mail
References

- Miran Lipovača, *Learn You a Haskell for Great Good!*
- Graham Hutton, *Programming in Haskell*
- O’Sullivan, Stewart, Goerzen, *Real World Haskell*
Grading

- 60% Machine Problems
- 20% Midterm Exam
- 20% Final Exam
Machine Problems

- 5-7 programming assignments in Haskell
- Submitting via shared, private Git repository on BitBucket
  - Invitations are coming soon!
For Friday

- Read Hughes’s “Why Functional Programming Matters”
- Start reading “Learn You a Haskell”
- Install the Haskell Platform (ideally, via Haskell Stack, haskellstack.org) & bring laptop to class