CS4450/7450: Review

Professor William L. Harrison

December 7, 2018

CS4450/7450: Principles of Programming Languages

- What is this course about?
  - Programming language Design

CS4450/7450: Principles of Programming Languages

- What is this course about?
  - Programming language Design
  - Data abstraction, Language Classes, Types
  - Explore these ideas by writing interpreters in Haskell

Grading

- Midterm (Friday, 10/5): 25%
- Written Homeworks + Programming: 40%
- Final: 35%

How to get an A in CS4450/7450

- When assigned reading, read it carefully.
  - Don't skim, *read*!

How to get an A in CS4450/7450

- When assigned reading, read it carefully.
  - Don't skim, *read*!
- When given a programming assignment, start early.
  - Last minute, start-the-night-before may work in other classes, but it's a prescription for disaster in CS4450.

How to get an A in CS4450/7450

- When assigned reading, read it carefully.
  - Don't skim, *read*!
- When given a programming assignment, start early.
  - Last minute, start-the-night-before may work in other classes, but it's a prescription for disaster in CS4450.
- Realize that you alone are responsible for your performance; be disciplined.

General Rules of Thumb

I.e., how I view tests

- Final will be comprehensive, although weighted towards post-midterm material.
- That being said, this course is cumulative.
- I don't typically ask "heavy" programming questions. That doesn't mean I won't ask programming questions, but I won't say "write an interpreter from scratch for X".

Course Materials

Both texts are online:

- *Learn You a Haskell for Good* by Miran Lipovaca
  Highly amusing and informative; available online.
- *Anatomy of Programming Languages* by William Cook. We covered Chapters 2-6. Did you read them?

Course Materials

Both texts are online:

- *Learn You a Haskell for Good* by Miran Lipovaca
  Highly amusing and informative; available online.
- *Anatomy of Programming Languages* by William Cook. We covered Chapters 2-6. Did you read them?
- Slides, of course.
  - Advice: Sit in a quiet room without distractions and read each set of slides. Do you understand what is said?
  - I say "without distractions" for a reason.

Course Materials

Both texts are online:

- *Learn You a Haskell for Good* by Miran Lipovaca
  Highly amusing and informative; available online.
- *Anatomy of Programming Languages* by William Cook. We covered Chapters 2-6. Did you read them?
- Slides, of course.
  - Advice: Sit in a quiet room without distractions and read each set of slides. Do you understand what is said?
  - I say "without distractions" for a reason.
- Code. If you look at the code from class and homeworks, do you understand what it does and why?

Course Materials

Both texts are online:

- *Learn You a Haskell for Good* by Miran Lipovaca
  Highly amusing and informative; available online.
- *Anatomy of Programming Languages* by William Cook. We covered Chapters 2-6. Did you read them?
- Slides, of course.
  - Advice: Sit in a quiet room without distractions and read each set of slides. Do you understand what is said?
  - I say "without distractions" for a reason.
- Code. If you look at the code from class and homeworks, do you understand what it does and why?
- Homeworks. If you lost points on a homework problem, do you know why? Do you understand the correct answer?

Language Syntax

- What are different forms of language syntax?
- What are different formalisms for representing language syntax?
- What is a parser?
- What is BNF? How is it used?
- Binding: What is static binding? Free variables?

Functional Programming Paradigms

BTW, this isn't a class on Haskell

- Data Types vs. Data Structures: what's the difference?
- Recursion and recursion patterns
  - left and right folds; maps; etc.
- Higher-order Functional Programming
  - Lambda expressions and "functions as values"
- Types for Functional Languages
  - Polymorphism, Higher-order, type classes
- Type-driven programming

Interpreters

- "Static" vs. "Dynamic" properties
- Variables, environments, closures
- Side-effects: using `Maybe` to model errors, state-passing views of imperative languages (e.g., Imp)
- "Front-end" issues
  - Read-Eval-Print loops
  - Static checking