Preliminaries

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

- lee@iit.edu
- http://moss.cs.iit.edu
- Office: SB 226C
- Hours: Wed & Fri 1-3PM, Zoom or In-person (by appointment only!)
TA: Nadeem Hussein

- nhussein@hawk.iit.edu
- Hours: TBA (on Discord)
Agenda

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

- I assume you are …
  - fluent in some programming language
  - familiar with procedural & OO paradigms
  - comfortable with development processes:
    - compilation, debugging, testing
CS 340: Programming Paradigms and Patterns

Announcements

- Welcome to the Spring 2023 edition of CS 340!

Calendar

Please note that readings for a given lecture should ideally be reviewed before coming to class, and will likely need to be re-read afterwards. Most readings are from Learn You a Haskell (LZh) and Real World Haskell (RWHS). All materials can be found online, and are linked below. Lecture notes can be found in the lecture repository.

The lecture calendar is tentative and may be updated.

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
<th>Reading(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 11</td>
<td>Syllabus and Course overview</td>
<td>slides-cs340.ppt</td>
<td>Syllabus</td>
</tr>
<tr>
<td>Jan 13</td>
<td>Functional programming</td>
<td>slides-lp.ppt</td>
<td>&quot;Why FP Matters&quot;</td>
</tr>
<tr>
<td>Jan 16</td>
<td>Haskell Language Overview</td>
<td>Let's Play!</td>
<td>LZh chapters 1 and 2 (skim)</td>
</tr>
<tr>
<td>Jan 20 - Jan 25</td>
<td>Types and Type Classes</td>
<td>LZh02.ppt</td>
<td>LZh chapter 3</td>
</tr>
<tr>
<td>Jan 27</td>
<td>Functions</td>
<td>LZh03.ppt</td>
<td>LZh chapter 5</td>
</tr>
<tr>
<td>Feb 1 - Feb 3</td>
<td>Lists</td>
<td>LZh04.ppt</td>
<td>LZh chapter 5 (ListIntro)</td>
</tr>
<tr>
<td>Feb 8</td>
<td>Testing and QuickCheck</td>
<td>LZh05.ppt</td>
<td>QuickCheck manual (skim)</td>
</tr>
<tr>
<td>Feb 10 - Feb 17</td>
<td>Recursion</td>
<td>LZh06.ppt</td>
<td>LZh chapter 5</td>
</tr>
<tr>
<td>Feb 22 - Feb 24</td>
<td>Higher Order Functions</td>
<td>LZh07.ppt</td>
<td>LZh chapter 5</td>
</tr>
<tr>
<td>Mar 8</td>
<td>Midterm Exam</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Course website: [http://moss.cs.iit.edu/cs340](http://moss.cs.iit.edu/cs340)
Blackboard: http://blackboard.iit.edu
Discord: TA office hours, class discussion, and Q/A (invite on course website)
Video playlist (on YouTube)
Primary text: “Learn You a Haskell for Great Good!”
References

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

- 50% Machine Problems
  - 4-6 Haskell programming assignments
- 25% Midterm Exam
- 25% Final Exam (Cumulative)
Late Policy

- 7-day late pool, distributed however you like across labs (a day at a time)
- If you’re out of late days, late submissions will not be accepted!
Exams

- Midterm tentatively scheduled for March 3
- Scores may be linearly scaled so that median/mean (whichever lower) is 75%
A: $\geq 90\%$
B: 80-89%
C: 70-79%
D: 60-69%
E: < 60\%
§ “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
Why Functional?

- Very different set of operating assumptions from your (likely) first model, imperative programming
  - No state mutations $\rightarrow$ referential transparency
  - Arguably easier to reason about (rigorously) and use for concurrency
  - You’ll read a paper on this for Friday!
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
E.g., Imperative & OOP patterns

- Loops/Iterators for array, list, or collection traversal
- Encapsulation with setter/getter methods
- Singleton & Factory patterns
- Observer pattern, aka Publish/Subscribe
“Gang of Four” book
Our focus: Functional patterns

- Structural and Generative recursion
- Higher order functions
- Functors and Monads
- Monoids and Foldables
- Etc.
Haskell

- Our functional language of choice: **Haskell**
- **Pure**: purely functional; side-effects are isolated
- **Strongly typed**: types are checked/enforced 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
  - Which will help in later classes and your career
Topics (not exhaustive)

- Functional programming
- Haskell Types and Typeclasses (like OOP on steroids)
- Higher Order Functions
- Functors and Monads
- Automated Property-Based Testing
- Concurrency and Software Transactional Memory
For Friday

- Read Hughes’s “Why Functional Programming Matters” (at least sections 1 & 2, if you can get further, great!)
- Start reading “Learn You a Haskell”