# Preliminaries

	nce Science
Science	Computer Science

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





# Agenda

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





# § Administrivia





# Michael Lee

- Email: <u>lee@iit.edu</u>
- Homepage: <u>http://moss.cs.iit.edu</u>
- Office: SB 226C
- Hours: Tue & Thu 10AM-12PM over Zoom (by appointment only!)





# Prerequisites

- I assume you are ...

- fluent in some programming language
  - familiar with procedural & OO programming
- 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 revisited afterwards. Most readings are from Learn You a Haskell (LYH) and Real World Haskell (RWH). 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.

Date	Торіс
Jan 11	Syllabus and Course overview
Jan 13	Functional programming
Jan 18	Haskell Language Overview
Jan 20 - Jan 25	Types and Type Classes
Jan 27	Functions
Feb 1 - Feb 3	Lists
Feb 8	Testing and QuickCheck
Feb 10 - Feb 17	Recursion
Feb 22 - Feb 24	Higher Order Functions
Mar 3	Midterm Exam

🔒 moss.cs.iit.edu

Ø 🔊

Ô + Ô

Notes	Reading(s)
slides-intro.pdf	<u>Syllabus</u>
slides-fp.pdf	"Why FP Matters"
Lect01.lhs Getting started Video	LYH chapters <u>1</u> and <u>2 (skim</u> )
Lect02.lhs	LYH chapter <u>3</u>
Lect03.lhs	LYH chapter 4
Lect04.lhs	LYH chapter 2 (List intro)
Lect05.lhs	QuickCheck manual (skim)
Lect06.lhs	LYH chapter 5
Lect07.lhs	LYH chapter 6

### Course website: <u>http://moss.cs.iit.edu/cs340</u>







### Blackboard: <u>http://blackboard.iit.edu</u>

blackboard.iit.edu     Content     C	+ C
ES-340-V01) Content Tools ∨ Partner Content ∨ ES ⓒ De held via Zoom (meeting ID 834 4004 4386) every Wednesday and Friday, 3:35-4:50PM cordings will be posted within 24 hours of each lecture to the class YouTube playlist.	<b>ΟΝ</b>
nents V Tools V Partner Content V 25 🔊 26 held via Zoom (meeting ID <u>834 4004 4386</u> ) every Wednesday and Friday, 3:35-4:50PM cordings will be posted within 24 hours of each lecture to <u>the class YouTube playlist</u> .	^↓
nents V Tools V Partner Content V Tes So	^↓
nents Tools   Partner Content  Is Coll Is held via Zoom (meeting ID 834 4004 4386) every Wednesday and Friday, 3:35-4:50PM cordings will be posted within 24 hours of each lecture to the class YouTube playlist.	τĻ
es held via Zoom (meeting ID <u>834 4004 4386</u> ) every Wednesday and Friday, 3:35-4:50PM cordings will be posted within 24 hours of each lecture to <u>the class YouTube playlist</u> .	
et held via Zoom (meeting ID <u>834 4004 4386</u> ) every Wednesday and Friday, 3:35-4:50PM cordings will be posted within 24 hours of each lecture to <u>the class YouTube playlist</u> .	
e held via Zoom (meeting ID <u>834 4004 4386</u> ) every Wednesday and Friday, 3:35-4:50PM cordings will be posted within 24 hours of each lecture to <u>the class YouTube playlist</u> .	





cs@iit	~	🗹 rules	Rules a
V INFORMATION	+		
🗹 rules	* ¢		
# announce			
# office-hours			
# welcome		@ M	ichael Lee elcome to
> GENERAL	+		
‡‡ mods		Ge	et help witl
$\ddagger$ mod-log		FI	edse redu a
~ CLASSES	+	1.	. Be kind a
‡≜ tas		2.	Keep nic
# cs100		(tl	hink PG-13
# cs115		3	. Don't spa
# cs116		4.	. No haras
# cs201			
# cs330		5. ha	No adver Ickathon ol
# cs331		int	formation
# cs340		6.	. Try to po
# cs350		cla	ass channe
# cs351		Cli	asses, and
# cs401		7.	Refer to t
# cs402		nt	tps://web
michaelee #0504	% n 🌣	₽ M	essage #rı

# Discord: TA office hours, class discussion, and Q/A (invite on course website)







•••	E < >	0		youtube.com	Ś				⊕ ₫	+	C
= •	NouTube	Search			0	Ļ	Ð		<u>ب</u>	Ð	
Home S Trending	"Applicatives" class (Function $f$ ) => Applied guide :: $a \rightarrow fa$ <*> :: $f(a \rightarrow b) =$ $f(a \rightarrow b) =$	diverfultere	=	SORT	Functors, Applic Michael Lee 03/25 CS 340 liv	atives, and re lecture	Monada	3			
Library	CS 340 lectur 14 videos • 665 views Unlisted 💌	PLAY ALL ITE VIDEOS * Last updated on Aug 27, 2020	=	What deser this mean for clasmoork? - 4. 4. 4. 4. 4. 4. 4. 4. 4. 4. 4. 4. 4.	Michael Lee 03/27 CS 340 liv Michael Lee	e lecture					
	No description		=		04/01 CS 340 liv Michael Lee	e lecture					
	Michael Le	e	=	And Andrewson and Antonia anto	04/03 CS 340 liv Michael Lee	e lecture					
			=	L Martin and A spectra for An and the second of the second WATCHED 24:26	State Monad Michael Lee						
				16. P.	04/08 CS 340 liv	e lecture					

### Video playlist (on YouTube)







### Primary text: "Learn You a Haskell for Great Good!"





# References

- Graham Hutton, Programming in Haskell
- O'Sullivan, Stewart, Goerzen, Real World Haskell

# - Miran Lipovača, Learn You a Haskell for Great Good!





# Grading

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









# Late Policy

- day at a time)
- If you're out of late days, late submissions will not be accepted!

### - 7-day late pool, distributed however you like across labs (a





## Exams

- Midterm tentatively scheduled for March 8
- Scores may be linearly scaled so that median/mean (whichever lower) is 75%

### uled for **March 8** ed so that median/mean





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 some language) is organized, expressed, or executed
- A given paradigm typically imposes some syntactic/semantic conventions or limits on programs
- E.g., procedural, imperative, object-oriented, functional
- We will be focusing on the **functional** paradigm





# Why Functional?

- Substantively different from the imperative paradigm, which is likely your "native" model
  - E.g., no state mutations  $\rightarrow$  referential transparency
  - Arguably easier to reason about rigorously in some contexts (and many other purported benefits)
- You'll read a paper on this for Friday!





# Pattern

- as possible to encourage reuse

- A reusable template for solving a common class of problem(s) - May be paradigm/language specific, and typically as abstract





# 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!
- way to think about and tackle problems
  - said language

- Learning a (different) new language gives you an entirely new

- Valuable, even if you don't actually code the solution up in





# A 'l'aste of 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
- - Which will help in later classes and your career



### - You want to learn new ways to reason about programming



# 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"



