CIS 552
Advanced Programming

Fall 2015
What is Advanced Programming?

• **Good** programmers get the job done

• **Excellent** programmers
  – write code that other people can understand, maintain and modify
  – rewrite/refactor code to make it clear
  – design *abstractions* to make it reusable
There are two ways of constructing a software design:

One way is to make it so simple that there are obviously no deficiencies,
and the other way is to make it so complicated that there are no obvious deficiencies.
The first method is far more difficult.
Simplicity through Purity

- Readable
- Reusable
- Modifiable
- Predictable
- Checkable

- Functional Programming: Focus on what code means instead of what it does
Absence of Unnecessary Elements

- No Mutation.
- No Objects.
- No IO.

Focus on what code *means* instead of what it does.
Simplicity through Abstraction

- Readable
- Reusable
- Modifiable
- Predictable
- Checkable

- Advanced type systems: Multiple levels of structure
CIS 552

Haskell
Why Haskell?

• Bleeding edge technology.
Why Haskell?

• Beautiful.
Why Haskell?

• Blows Your Mind.
Why Haskell?

• Fun.
Why Haskell?

• Types.
Why Haskell?

• Real.
Course content

Functional Programming
  – Black-belt Haskell
  – Many small-scale case studies
  – In-class design analysis

Advanced Programming Techniques
  – Modular design and abstraction
  – Black-belt types
  – Test driven development
  – Debugging
  – Collaboration (pair programming)

Lots of programming!
  – Weekly-ish assignments
  – Larger project at the end
What this course is not

• CIS 350/573, Software Engineering
  – Focuses on "Software in the large”
  – How to deal with code you didn’t write
  – Problems that arise in projects that are too large for one person
    • lifecycle models
    • project management
    • design modeling notations (UML)
    • formal specification

• The two courses complement each other
Audience

• People with strong background in programming and mathematics
• No experience with FP expected
• Priority to CIS undergrads/grad students
• If not registered and not on the waitlist, send email (cis552@cis.upenn.edu) immediately
Course Staff

Instructor: Dr. Stephanie Weirich
sweirich@cis.upenn.edu

TA: Raul Martinez
mraul@seas.upenn.edu

Fill in instructor:
Dr. Benjamin Pierce
Grading

• 60% Programming assignments
  – mostly in pairs, randomly assigned
  – graded on correctness, style and (asymptotic) efficiency

• 20 % Final Projects (your choice)

• 20 % Class participation
  – questions – help me learn your name!
  – HW debriefings in class
  – Class activities
  – Piazza participation
  – Read a good blog post about FP? post about it on Piazza!
Where to go for help

• Course website:
  – http://www.seas.upenn.edu/~cis552
  – lecture code, homework, resources
• Textbook: Real World Haskell (free online)
• Class forum: piazza.com
• Office hours
  – listed on course web page
  – SCW: today after class
Things to do now

• Work on HW0 (due next Tuesday at midnight)
• Read at least Preface and Chapter 1 of Real World Haskell before next class
• Look for a message from Piazza later today
  – Email sweirich@cis.upenn.edu if you don’t see it
• If you are not currently registered, see me after class
Homework #0

• Available from course website
• Get everything set up on your machine
  – Haskell compiler and libraries
  – Version control software
  – Editor
• Download first lecture, fill in blanks, and run
• Submit via course website
• Due Tuesday, Sept 1\textsuperscript{st} at midnight
Homework #1

- Available today, covers next week's material
- Download/submit from course website
- Work with a partner (your choice), submit only once
- Must compile with –Wall –Werror to get any credit.
- Due Tuesday, Sept 8\textsuperscript{th} at midnight
- Late policy
  - 10 point penalty for up to 24 hours late
  - 20 point penalty for up to 48 hours late
  - no credit for assignments submitted after 48 hours
Homework style

• Style guide on course website: read it
• Interactive HW discussions in class
  – Examples of good / bad style
• Revise, revise, revise
  – Passing all the tests is not enough
  – Code must be effective technical communication
  – How can I make this code cleaner, more general, more clear about what it is doing?
• New this semester: peer style feedback
Academic Integrity

• We expect you to follow Penn's policies on Academic Integrity.

• Do not plagiarize or copy code
  – Using a library function is ok (and better style)
  – Looking up the source of that function and presenting it as your own work is not

• Do not get someone else to do your work for you
  – Ok to ask for help debugging a type error
  – Not ok to ask for the answer
  – Ask high-level questions on piazza
Lecture format

• Last powerpoint slide of the semester
• Bring your laptops and work along
• Interactive code demos and exercises
• Ask questions!
fin
So, Who Uses FP?
So, Who Uses FP?

Microsoft®
So, who uses FP?
So, Who Uses FP?
So, Who uses FP?
Goal: Obviously no deficiencies

• Want code that is so simple, it obviously works

simple code is

Readable
Reusable
Modifiable
Predictable
Checkable

• OK... so what makes code simple?