Aarti Gupta

David Walker

COS 326 Functional Programming: An elegant weapon for a more civilized age

Princeton University

slides copyright 2021 David Walker and Aarti Gupta
permission granted to reuse these slides for non-commercial educational purposes

Your professors

Aarti Gupta

David Walker

Your professors

Aarti Gupta

David Walker

Alonzo Church, 1903-1995
Princeton Professor, 1929-1967

In 1936, Alonzo Church invented the lambda calculus. He called it a logic, but it was a language of pure functions -- the world's first programming language.

He said:

"*There may, indeed, be other applications of the system than its use as a logic.*"

Indeed!

Alonzo Church
1934 -- developed lambda calculus

↓

Programming Languages

Alan Turing (PhD Princeton 1938)
1936 -- developed Turing machines

↓

Computers

Optional reading: ***The Birth of Computer Science at Princeton in the 1930s***
by Andrew W. Appel, 2012. http://press.princeton.edu/chapters/s9780.pdf

A few designers of functional programming languages

Alonzo Church:
λ-calculus, 1934

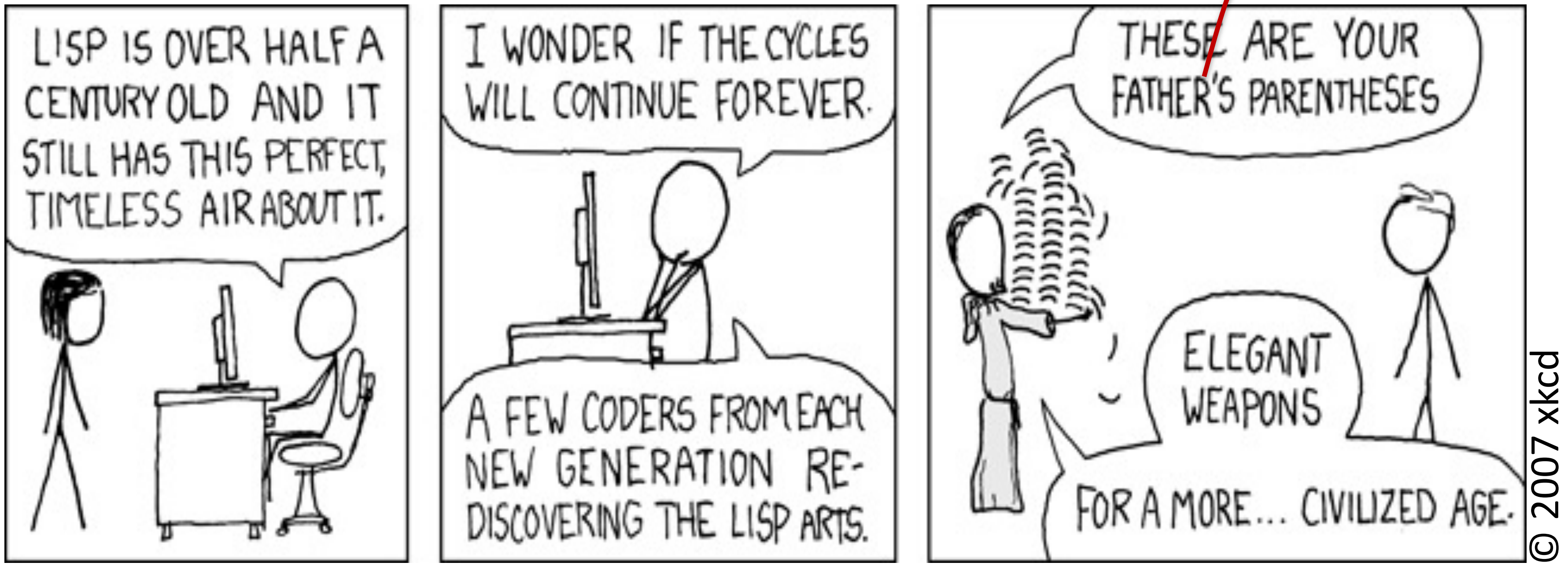
John McCarthy
(PhD Princeton 1951)
LISP, 1958

Guy Steele & Gerry Sussman:
Scheme, 1975

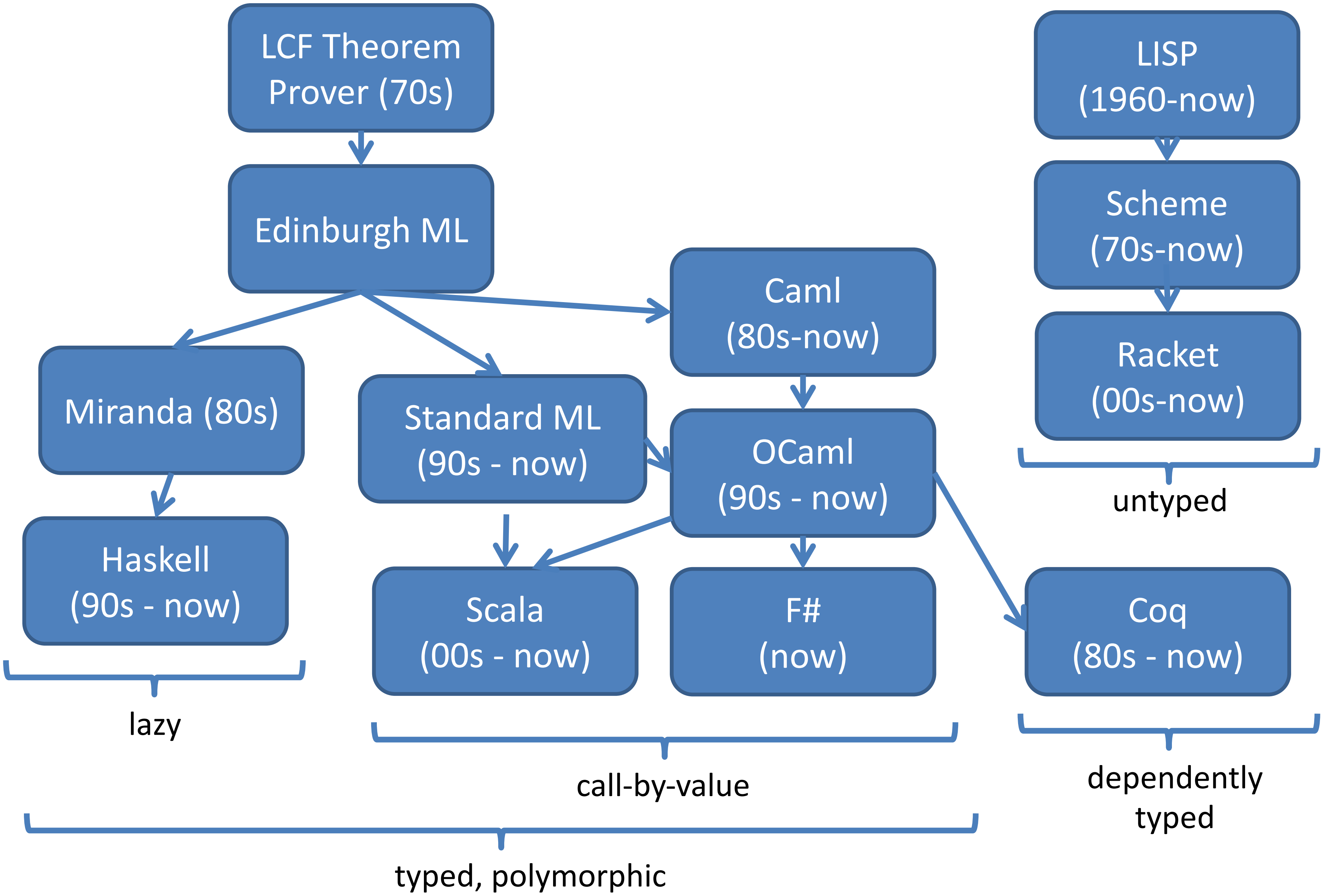
LISP, 1960

```
(define
    (my-max3 x y z)
    (if (and (> x y) (> x z))
         x
         (if (> y z)
             y
             z)))
```

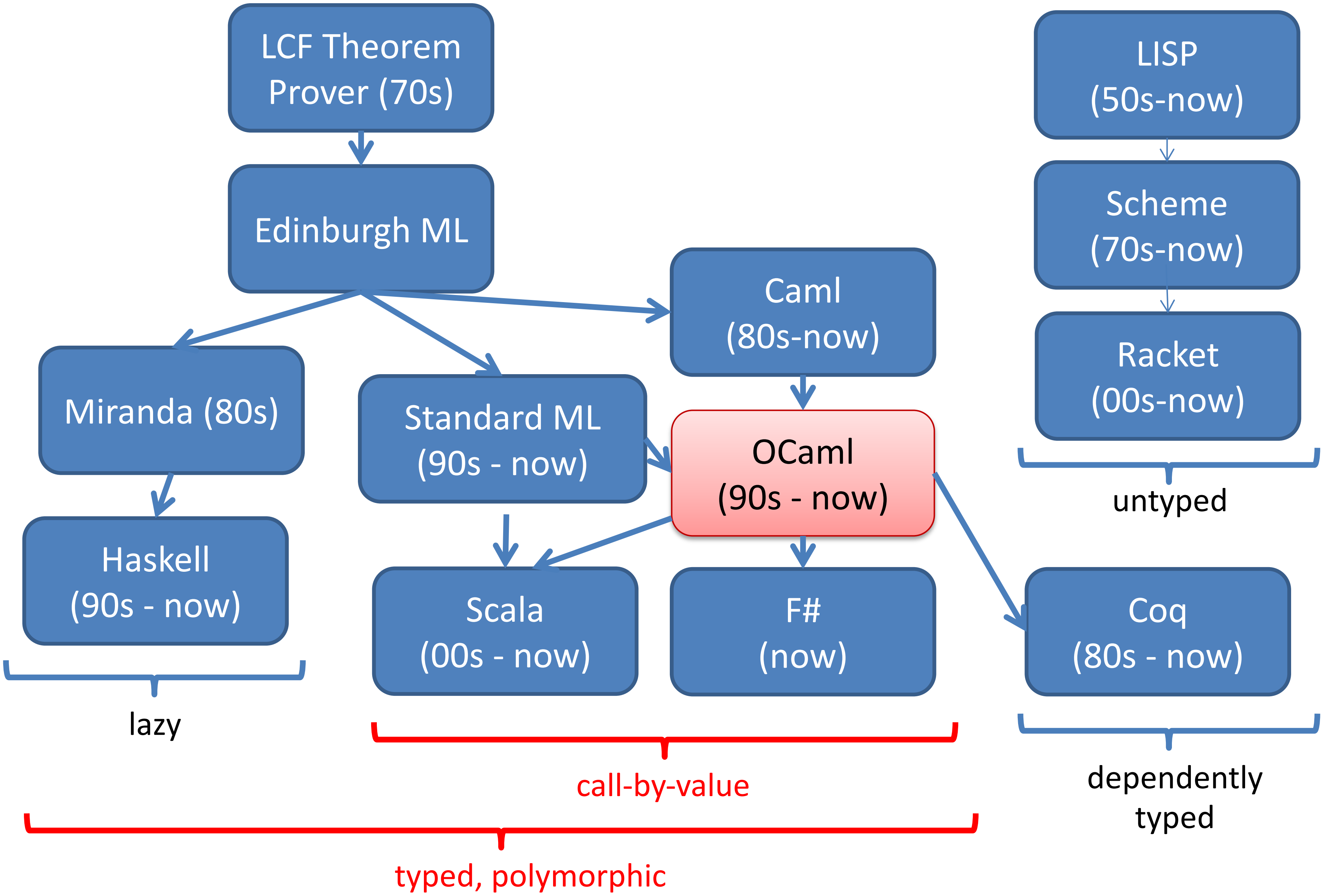
or mother's


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.
© 2007 xkcd

Vastly Abbreviated FP Genealogy



Vastly Abbreviated FP Geneology



Functional Languages: Who's using them?

twitter

map-reduce in their data centers

Google

Scala for
correctness, maintainability, flexibility

facebook

Erlang for concurrency,
Haskell for managing PHP,
OCaml for bug-finding

Microsoft Be what's next.

F# in Visual Studio

mathematicians

Coq (re)proof of
4-color theorem

bluespec

Haskell to
synthesize hardware

BARCLAYS

Haskell
for specifying
equity derivatives

www.artima.com/scalazine/articles/twitter_on_scala.html
www.infoq.com/presentations/haskell-barclays
www.janestreet.com/technology/index.html#work-functionally
msdn.microsoft.com/en-us/fsharp/cc742182
research.google.com/archive/mapreduce-osdi04.pdf
www.lightbend.com/case-studies/how-apache-spark-scala-and-functional-programming-made-hard-problems-easy-at-barclays
www.haskell.org/haskellwiki/Haskell_in_industry

COURSE LOGISTICS

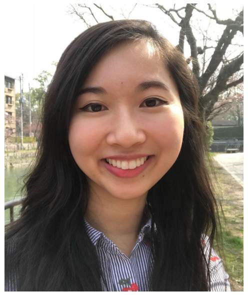
Course Staff

Professors

Aarti Gupta

David Walker

Preceptors


Kathy Chen

Anja Kalaba

Lisa Liu

Undergraduate Course Assistants: TBD (watch the course website)

Resources

- coursehome: http://www.cs.princeton.edu/~cos326
- Communication
  - Ed: https://us.edstem.org/courses/7418/discussion/
- Lecture schedule and readings:
  - $(coursehome)/lectures.php for schedule and slides
  - Lecture material is available in prerecorded videos
  - Join the professors during lecture meeting time
    - Mon and Wed: 11am – 12:20pm ET, McCosh Hall 46
    - Part 1: Q&A, please watch the video material in advance
    - Part 2: Ask-me-anything sessions *(new! see next slide)*
- Assignments:
  - $(coursehome)/assignments.php
- Precepts: (one hour per week, on Th/Fri)
  - Precept attendance is mandatory.
- Install OCaml: $(coursehome)/resources.php

Lectures and Ask-me-anything (AMA) Sessions

- Each lecture class has two parts
  - Part 1: Ask the professor any questions on the ***lecture*** material
    - Questions on assignments are better left to office hours and precepts
  - Part 2: **Ask-me-anything (AMA) session**
    - When the first part questions are done, the AMA starts
    - The main idea is to get to know each other, would be nice to reconnect after the pandemic
- AMA
  - We plan to assign ~10 students to each AMA (list on website)
  - Assigned students can ask *any* question
    - E.g., How do I come up with an IW topic? What's the coolest idea in computer science you've come across? What non-computer science class would you take?
    - More ideas are welcome! (but we'd like to steer clear of politics)
  - We too might ask questions, to get to know you and get feedback

JRW 326 Course

- Also new this year, we have a Junior Research Workshop (JRW) course associated with COS 326

- The JRW 326 course is taken *only* by COS AB students
  - It serves as their IW requirement in Junior Fall semester
  - Some material in JRW 326 will be based on the material we cover in the lectures and precepts here

A Typical Week

Monday

- Lecture meeting at 11-12:20, with professor

Wednesday

- Lecture meeting at 11-12:20, with professor

Thursday

- Assignment due (in many weeks)

Thursday/Friday

- Mandatory precept reinforces lecture content in small groups
- You may have questions for your preceptor about the *next* assignment

Collaboration Policy

The COS 326 collaboration policy can be found here:

http://www.cs.princeton.edu/~cos326/info.php#collab

Read it in full prior to beginning the first assignment.

Please ask questions whenever anything is unclear, at any time during the course.

Sample README.txt (abridged)

Netids (include all members of group, if parternering):
[list here]

In doing this homework I used the following sources:

1. Sources I don't need to mention [see notes 1 and 4]
2. Authorized sources [see notes 2 and 4]
   [list here]
3. Unauthorized sources [see notes 3 and 4]
   [list here]

This paper represents my own work in accordance with University regulations.

Signed, [your name(s):]

NOTE 1: Sources you don't need to mention
this semester's lectures and precepts, the course web site, the assignment handout (download), Real World OCaml, and the OCaml manual.

NOTE 2: Authorized sources include:
professors and preceptors, advice from other students (but not looking at their solutions); other books, and (within reason) web sites such as stackoverflow.com.

NOTE 3: "Why would I list an unauthorized source?"
Using an unauthorized source without citing it is an Academic Violation under Princeton University's disciplinary code, and can result in suspension from the University.

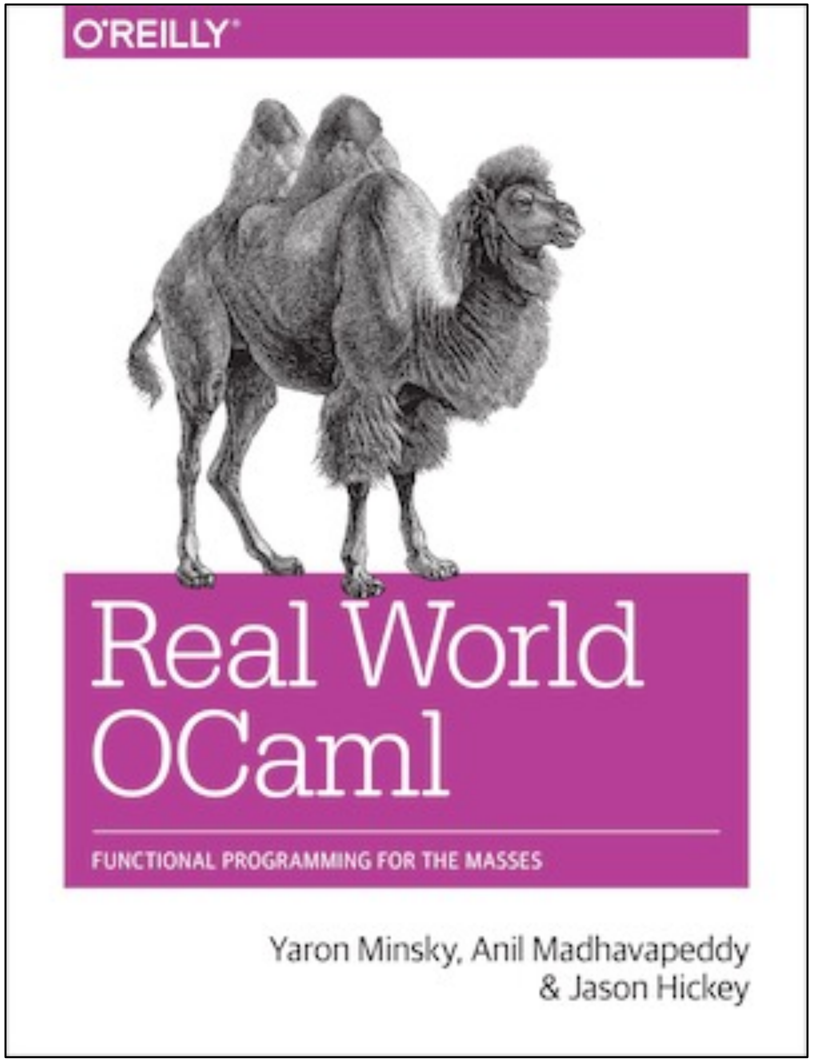
Using an unauthorized source and citing it clearly is "merely" a violation of this homework's instructions, and can result in (at most) getting a zero on this homework.

Unauthorized sources include, at least: other people's solutions to these (or similar) homework problems.

NOTE 4. If you paste in code from from these sites, you should clearly cite it at the point of use, in accordance with Section 2.4.6 of RRR for "direct quotation or extensive paraphrase".

Please limit the amount of this that you do in accordance with the principle that the purpose of these homeworks is so that you can learn how to do things yourself.

Course Textbook


O'REILLY

Real World OCaml

FUNCTIONAL PROGRAMMING FOR THE MASSES

Yaron Minsky, Anil Madhavapeddy & Jason Hickey

http://realworldocaml.org/

Exams

Two take-home exams

- Exam 1 during midterm week, Oct 12-13
- Exam 2 near the end of classes, Nov 30-Dec 1

No Final Exam!

Assignment 0

Download and install OCaml and get emacs or your favorite editor set up to process OCaml code (syntax highlighting; type checking)

See the Resources Page for install instructions on your platform:

http://www.cs.princeton.edu/~cos326/resources.php

Thinking Functionally

pure, functional code:

```
let (x,y) = pair in
(y,x)
```

you *analyze* existing data (like pair) and you *produce* new data (y,x)

imperative code:

```
temp = pair.x;
pair.x = pair.y;
pair.y = temp;
```

commands *modify* or *change* an existing data structure (like pair)

Thinking Functionally

pure, functional code:

```
let (x,y) = pair in
(y,x)
```

- *outputs are everything!*
- *output is <u>function</u> of input*
- *data properties are stable*
- *repeatable*
- *parallelism apparent*
- *easier to test*
- *easier to compose*

imperative code:

```
temp = pair.x;
pair.x = pair.y;
pair.y = temp;
```

- *outputs are irrelevant!*
- *output is not function of input*
- *data properties change*
- *unrepeatable*
- *parallelism hidden*
- *harder to test*
- *harder to compose*

This simple switch in perspective can change the way you

think

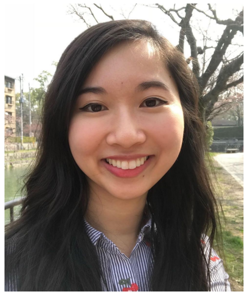
about programming and problem solving.

Have fun!

Let's make this an amazing semester!

Aarti Gupta

David Walker


Kathy Chen

Anja Kalaba

Lisa Liu