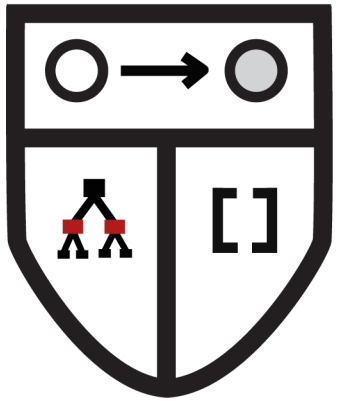

CS 3110

Victory Lap

Prof. Clarkson
Fall 2015

Today's music: "We are the Champions" by Queen

Victory Lap

Extra trip around the track by the exhausted victors (us) ☺

Thank you!

Huge thank you to TAs and consultants!

Quinn Beightol, Aditya Bodas, Patrick Cao, Jonathan Chan, Jake Chen, Jonathan Chen, Alan Cheng, Yogisha Dixit, Daniel Donenfeld, Trevor Edwards, Tyler Etzel, Seung Hee Han, Amber Hillhouse, Remy Jetty, Akekawit Jitprasert, Chaein Jung, Emmett Kotlikoff, Sushmitha Krishnamoorthy, Daniel Liang, Eric Lin, Genki Marshall, Leo Mehr, Alice Meng, Giang Nguyen, Luke Nicholson, Eric Pass, Linda Pei, Charles Qian, Niranjan Ravi, Ralph Recto, Daniel Sainati, Nimit Sohoni, Lucia Song, Yuxiao Tan, Ross Tannenbaum, Shiyu Wang, Richard Wu, Guandao Yang, Anna Yesypenko, Liang Zhang, Richard Zhang

And recitation instructor Prof. Mike George!

Thank you!

Thank you to Piazza heroes!

Top Student Contributors

Chirag Bharadwaj 427 contributions; 104 days online

Gur-Eyal Sela 165 contributions; 100 days online

Jimmy 165 contributions; 89 days online

Risa Wenhui Feng 154 contributions; 85 days online

Albert Zhang 153 contributions; 91 days online

Thank you!

And a huge thank you to all of **you!**

- You surmounted a daunting challenge
- You occasionally laughed at my dumb jokes ☺

I <3 this course. You make it all worthwhile.

What did we learn?

- You feel exhausted...
- You're tired of coding...

...step back and think about what happened along the way

What did we learn?

From the syllabus:

- Functional programming
- Writing and using specifications
- Modular programming and data abstraction
- Reasoning about program correctness
- Reasoning about system performance
- Useful and efficient data structures

...and some cross-cutting, big ideas...

1. Languages can be learned systematically

- Every language feature can be defined in isolation from other features, with rules for:
  - syntax
  - static semantics (typing rules)
  - dynamic semantics (evaluation rules)

- Divide-and-conquer!
- Entire language can be defined mathematically and precisely
  - SML is. Read *The Definition of Standard ML (Revised)* (Tofte, Harper, MacQueen, 1997).

- Learning to think about software in this "PL" way has made you a better programmer even when you go back to old ways
  - And given you the mental tools and experience you need for a lifetime of confidently picking up new languages and ideas

2. Immutability is an advantage

- No need to think about pointers or draw memory diagrams
- Think at a higher level of abstraction
- Programmer can alias or copy without worry
- Concurrent programming easier with immutable data
- But mutability is appropriate when you need to model inherently state-based phenomena
  - or implement some efficient data structures

3. Programming languages aren't magic

- Interpretation of a (smallish) language is something you can implement yourself

- Domain specific languages (DSL): something you probably *will* implement for some project(s) in your career

4. Elegant abstractions *are* magic

From a small number of simple ideas...

...an explosion of code!

- map and fold
- lists, trees, and dictionaries
- monads: Async

5. Building software is more than hacking

- **Design**: think before you type
- **Empathy**: write code to communicate
- **Assurance**: testing and verification
- **Performance**
- **Group work**

6. CS has an intellectual history created by people like you

Big ideas

1. Languages can be learned systematically
2. Immutability is an advantage
3. Programming languages aren't magic
4. Elegant abstractions are magic
5. Building software is more than hacking
6. CS has an intellectual history created by people like you

What next?

- Follow-on courses:
  - CS 4110 Programming Languages and Logics (how to define and reason about programming languages)
  - CS 4120 Compilers (how to implement programming languages)

- Join the course staff?
  - CS department collects applications
  - Apply in May 2016 to be on my staff for Fall 2016

- Stay in touch
  - Tell me when 3110 helps you out with future courses (or jobs!)
  - Ask me cool PL questions
  - Drop by to tell me about the rest of your time in CS (and beyond!)… I really do like to know

- DO AMAZING THINGS WITH YOUR LIFE

DJ
#flOCaml

Q&A

Final Exam

- Saturday, 12/12/15, 2:00-4:30 pm, Olin 155, 165, 255
- Covers everything in the course
- You may have **three** pages of notes
- (remaining details will be posted on Piazza)

Final Grades

- Fill out the course eval to get your +1%
  - Take time on this, especially the free response questions
  - Fill out the Clarkson course eval; don't worry about the George course eval
- Final grades:
  - uploaded to registrar approx. 24 hours after final exam scores are released on CMS
  - less than 24 hours after that, the registrar locks them, and we can't change them

Finally

- The most important idea of this course:
  - complicated artifacts can be broken down into small pieces
  - you can then study those small pieces and understand how they work in isolation
  - then you can understand why their aggregation achieves some goals

- Examples: the OCaml language, or a module you designed

- That kind of analysis is applicable anywhere, not just programming

Upcoming events

- [today] A6 due (including Project Implementation), but no late penalty up to the hard deadline

This is ...

This is victory.

THIS HAS BEEN 3110

Music: End credits from Final Fantasy IX