Meeting 27: Retrospective, Subtyping Polymorphism

HW5: Time: 16.6 hours avg, 9.5 stddev, 14 median; Hardness: 5.0 (out of 6) avg, 1.0 stddev, 5 median
HW6: Time: 10.6 hours avg, 6.2 stddev, 10 median; Hardness: 4.3 (out of 6) avg, 1.1 stddev, 4 median

HW6 Comments
- Not a difficult assignment overall. Good to slow down to see the "big picture". Things have come together to a great extent.***
- A lot more manageable.
- Pretty straightforward.
- This assignment was easy and short.
- "I really enjoyed this class. It was challenging but I think I did learn the material."

HW6 Hard Parts
- Understanding memory. Thinking about addresses.
- Working with tuples.
- Had to read typing judgments a couple times.
- "I thought the hardest part was reasoning about things like call-by-name, call-by-reference, etc., but the operational semantics and typing rules made these things more or less straightforward."
- "This one I found easier than others. I think that's mostly because I'm understanding judgements much better. Because of this, implementing the code is really just translating the syntax of the judgements into the syntax of scala"
- Writing Smalla programs.***

About Your Classmates
- A great dancer
- Of Polish decent
- Related to Abraham Lincoln
- Loves home brewing
Logistics for Thu/Fri Presentations

- Teams of 4
- 8 min each
- Run continuous alarm
- Next group “on deck”
Course Goals

You will be able to learn new languages quickly and select a suitable one for your task.

- Diffr between the syntax versus semantics of a language?

let x = 1 in x

let y = 1 in y

syntax: grammars, abstract syntax, d-equiv, ambiguity, higher-order abstract syntax, names/binding/scope, associativity, precedence

semantics: judgments/inference rules, evaluation
- static versus dynamic typing
- base types: ints, booleans
- aggregate data struct records, unions
  [objects, structs]
- recursion
- mutation: variables, dyn mem alloc
  - proc abstraction = param passing
  - eager versus lazy evaluation
  [objects + dynamic dispatch]
You will gain new ways of viewing computation and approach algorithmic problems.

- Function can be values that can be stored and passed

\[ \Rightarrow \text{ methods (in OO)} \]

JavaScript

```javascript
0.m = function() {
    ...
}
```

we have methods in smaller
& You will gain new ways of viewing programs

- A program is algebraic object that can be manipulated
- Interpret a program by "reducing"/"simplifying" step-wise to a value
- Tricky to write programs that manipulate programs

& You will gain insight into avoiding mistakes when you design languages

- Dynamic scoping
- Judgements/Inference Rules