CSE 114A

Foundations of Programming Languages

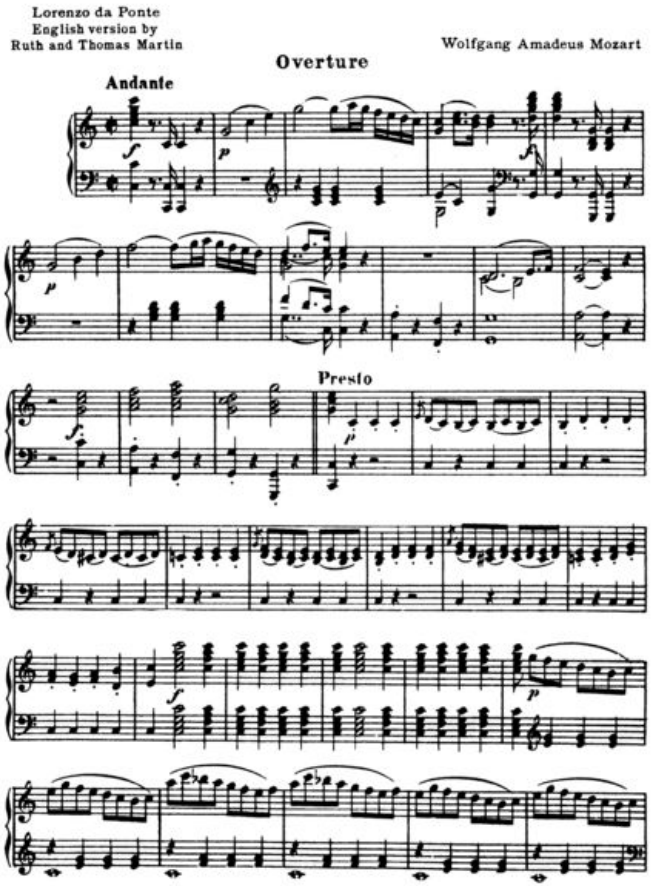
Lecture 1: Course Overview

Based on course materials developed by Ranjit Jhala and Owen Arden

So why study PL ?

Programming language shapes Programming thought

Learn New Languages/Constructs



New ways to:

- describe
- organize
- think about

computation

Course Goals

"Free your mind"

-Morpheus

Imperative Programming

$$x = x+1$$

$$x = x+1$$

Imperative = Mutation

Imperative = Mutation

Bad!

Don’t take my word for it

John Carmack

Creator of FPS: Doom, Quake,...

Don’t take my word for it

Tim Sweeney (Epic, Creator of UNREAL)

“In a concurrent world, imperative is the wrong default”

Functional Programming

Functional Programming ?

No Assignment.

No Mutation.

No Loops.

OMG! Who uses FP?!

So, Who Uses FP ?

MapReduce

So, Who Uses FP ?

Microsoft®

Linq, F#

So, Who Uses FP ?

Erlang

So, Who Uses FP ?

twitter

Scala

So, Who Uses FP ?

Wall Street

(all of the above)

So, Who Uses FP ?

...CSE 114A

Course Mechanics and Logistics

Logistics

Course website:

https://ucsc-cse-114a.github.io/Winter23/

Resources

Course texts (optional):

- An Introduction to Functional Programming Through Lambda Calculus by Greg Michaelson. Free pre-print.
- Thinking Functionally with Haskell by Richard Bird. Available online (free via library).
- Programming in Haskell (2nd ed.) by Graham Hutton.
- Real World Haskell by Bryan O'Sullivan. Available online (free via library).
- Learn You a Haskell for Great Good by Miran Lipovača. Available free online
- Write You a Haskell by Stephen Diehl. (incomplete, but useful) Available free online

Recommended IDE: VS Code

- Nice IDE setup for Haskell
  - Devcontainer: A Haskell dev environment is built in a docker container
  - VS Code automatically mounts the container volume

Peer Instruction (ish)

Peer Instruction

- Make class interactive
  - Help YOU and ME understand whats tricky
- Respond to in-class quizzes
  - 5% of your grade
  - Respond to 75% questions
- Bring laptop/phone if you have one

In Class Exercises

1. Solo Vote: Think for yourself, select answer

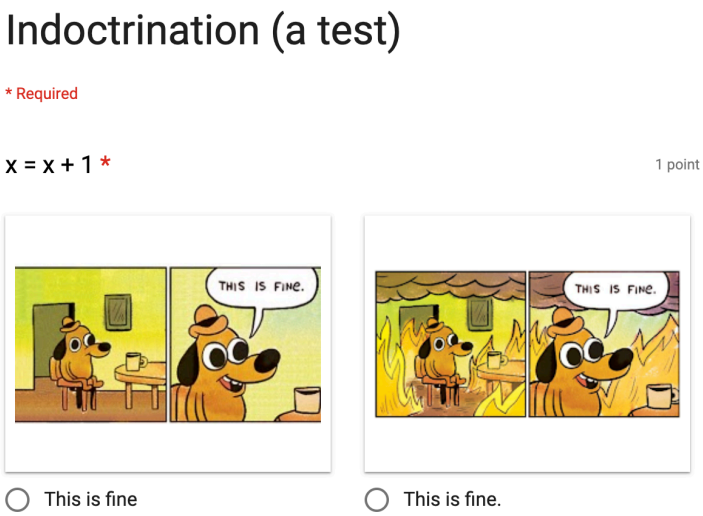
2. Discuss: Analyze Problem with neighbors
   - Practice analyzing, talking about tricky notions
   - Reach consensus
   - Have questions, raise your hand!

3. Group Vote: Everyone in group votes

4. Class-wide Discussion

In Class Exercises

Let’s try it out (if you have a device):

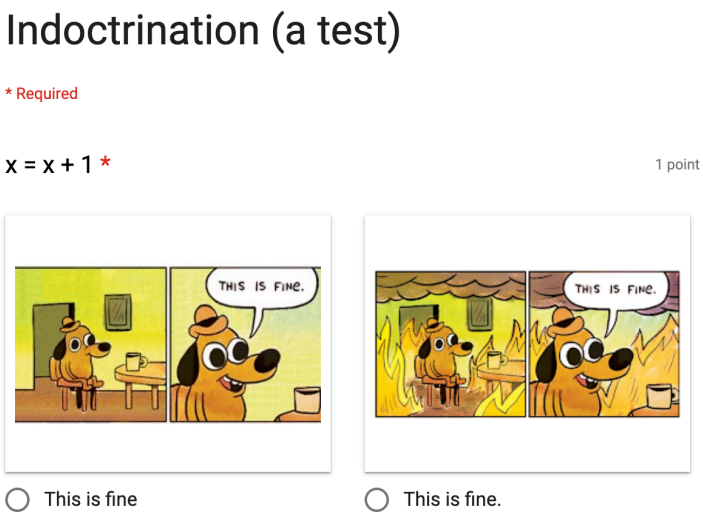


http://tiny.cc/cse116-trial

Make your individual choice

In Class Exercises

Let’s try it out (if you have a device):



http://tiny.cc/cse116-trial

Now “confer” with a neighbor and agree on a choice for your group

Requirements and Grading

- In-Class Exercises: 5%
- Midterms: 30%
- Programming Assignments (6): 30%
- Final: 35%

Resources

- Online lecture notes
- Readings and exercises
- Lectures recorded on Yuja
- Discussion sections
- TA and Tutor Office hours

Programming Assignments

All assignments are managed through GitHub Classroom (link on course page).

- **You must *push* your submitted code.**

Deadline Extension:

- Eight “late days”, used as “whole unit”
- 5 mins late = 1 late day
- Plan ahead, **no other extensions**

See course webpage for HW deadlines

Programming Assignments

Unfamiliar languages

\+ Unfamiliar environments

Start Early!

Weekly Programming Assignments

Scoring = Test suite

No Compile, No Score

Weekly Programming Assignments

- Programming Assignments done **ALONE** or in **(official) groups of two** (as permitted)
- We use plagiarism detection software
  - MOSS is fantastic, plagiarize at your own risk
- **Zero Tolerance**
  - offenders punished ruthlessly
- Please see academic integrity statement:
  - https://ue.ucsc.edu/academic-misconduct.html

Weekly Programming Assignments

Forget Java, C, C++ …

… other 20th century PLs

Don't complain

… that Haskell is hard

… that Haskell is @!%@#

Immerse yourself in new language