```
;; Question 1:
;;
;; How many hours between now and the final do you plan to study for it?
;; (All answers will be marked correct, answer truthfully.)
;;
;; A.  5
;; B. 10
;; C. 15
;; D. 20
;; E. 25 or more
```

```
;; Question 2:
;;
;; I know that:
;;
;; A. working through practice problems on my own is the best way to
;;    prepare for the final
;; B. WATCHING video is nowhere near as effective as playing a little,
;;    stopping, trying to get ahead, restarting and comparing what I
;;    did to the video
;; C. Office hours are the best place to work through practice problems
;;    because I can get hints from course staff rather than looking at solution
;; D. Using the problem bank is a good way to assess what material I already
;;    have mastered and what material I should be working on
;; D2. The exam will have more than one problem, spending too much mental energy
;;    preparing for tandem worklist graph problems isn't a good use of my time
;;    until I am already comfortable with everything up to that point
;; E. All of the above
```

```
;; Question 3:
;;
;; A. Next fall I hope to not take any more online courses at all.
;; B. Next fall I would like all my courses to be online.
;; C. Next fall I would like the option of taking one or two courses online.
```

Lots of office hours will be posted.

Instructor hours will be posted shortly.

There are comments about the final in this lecture, but no “hidden hints”.

The final covers the whole course.

Faculty In Your Future CS Courses

- A diversity of specializations: programming languages, software engineering, CS education, systems, graphics, HCI, machine learning, algorithms, numerical computation… and more
- And lots of other diversity too
    - black, Asian… gay, straight… cisgender, trans… native, immigrant…
- Respect every one of them
- They are great people and great teachers

Plan for rest of today

- What about other languages?
- What have you learned?
- How do we feel about the final?
- What's next?
- Student Survey

Other languages

- Everything you've learned in 110 works in other languages
  - data design, function design, tests, templates…
  - above all, working systematically to narrow the gap between problem and solution

- Learn new languages by reading code
  - find code that "must do X"
  - use what you know to understand the chunks (the templates)
  - figure it out from there

Our last starter…

```
//@template (listof X) w/ acc
def fn(lox):
    res = ...
    for x in lox:
        res = res...x
    return ...x
```

What have you learned ... about design?

- Figuring out what you actually want is half the battle

  - signature
  - purpose
  - examples (wrapped in check-expect)

  - information examples
  - interpretation

What have you learned ... about design?

- then the structure of the solution
  - template origins
  - accumulator types and invariants
- and the details
  - fill in … according to all above
  - debug

```
All 5 tests pass!
```

What have you learned ... about design?

- but sometimes

```
50% of  50%   Submitted tests: correct - all submitted test pass.
 0% of  50%   Additional tests: incorrect - 3 autograder internal additional tests failed.
```

  - despite your best efforts
  - what you end up with is not what you really wanted
  - go back and systematically revise the design, and learn from that error

Hopefully you also learned

- that you can solve larger and harder problems than you thought
- some of what it will take to solve harder and harder problems

- patience, attention to detail, humility are not small parts of it

The Recipes are about

- Making systematic progress, bit by bit, to solve a complex problem.
- Write down the easiest thing first
  - what is information in problem domain, form of information, type comment, examples, template
  - fn name, signature, purpose, examples, template, code rest of function body
- Don't jump to a solution too soon
  - information, form, DD, signature, purpose, template, fill in … vs. just start coding
- Describe the goal in different forms.
  - types (signature), text (purpose), examples, template
- Work it out, piece by piece, always writing down what you just figured out.
- Using software engineering and computer science to structure the solution into separate parts
  - What's the information? What's the changing information?
  - What kind of problem is it? (Simple functional, World, Search…)
  - What's the basic function strategy? (structural, generative, search...)
  - What are the templates?

“We are what we repeatedly do. Excellence, then, is not an act, but a habit.”

- Many

“Chance favours the prepared mind.”

- Louis Pasteur

What have you learned ... about Computer Science?

- foundations of software engineering
  - much more in 210, 310, 410
- simple functional programming language
  - much more in 311, 312, 411
- a bit about algorithms and data structures
  - trees, graphs, sorting, searching
  - much more in 221, 320, 420
- a very little bit about systems and architecture (MVC)
  - much much more in 213, 313, 317

- but there's much much much much more beyond that

203	Non-majors 2nd course in programming NEW		
330	Non-majors machine learning NEW		
210, 310, 410	Software Engineering	SE/PL	
311, 411	Programming languages		
213, 313, 317, 415	Systems…	Systems	
304, 404	Databases		
121, 221, 320, 420	Algorithms, data structures, theory	Theory	
302, 303	Numerical computation	+inf.0	
322, 340	AI and machine learning	speech, vision, “you may like”…	
314, 424	Graphics	movies	
344, 444	Human Computer Interaction	how do people interact with tech…	
436	Visualization	presenting complex information	
…			
DSCI	Data science		

Ask your TAs!!!

Jobs outside universities…

- Build your own portfolio, get internships, COOP and/or work with faculty
- Interviews are about what you can DO, not what you remember

- A reference letter that helps must have all three of these properties:
  - comes from someone the committee will trust
  - comes from someone who has something specific to say about you
  - says specific things about you that compare favorably with rest of group

Final review

- Monitor Piazza forum closely – ask well setup questions
- Work through problems (as opposed to reviewing solutions)
- Focus on WHY you type the code you do
  - What rule? What observation? What idea?

- Work through them online
- Practice getting RUNNING solutions
- Work through being stuck, don't just look at the solution
- Work in office hours (hints are better than looking at solutions)
- Work in 2.5 hour chunks w/o interruption

- Several hours every day beats 24 hours the day before the exam

You are now bearers of IT sorcery…

- Information technology is extraordinarily powerful
  - DeepMind protein folding
  - Self driving cars
  - …
  - UBC SSC
- Try to do good things with that power
  - not everyone can work on cancer (former head TA S*** in NYC working on that now)
  - not everyone can go to cool hot company (former students are at all of them)
  - we also need payroll systems, and traffic automation, and 10k+ other important systems
  - but some things are less good…, some questions to ask
    - where does the wealth go? where does it come from?
    - who has the power over the information?
    - who has agency in how the system works?