Lecture 23

and then we were done

There will be clickers!
All answers will be correct, but not answering will not be correct.
Leave your devices open.

Plan for rest of today

- How do you feel about final?
- What have you learned?
- What about other languages?
- What’s next?
- A few thoughts on generative AI
- Student Survey

```
;; 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.

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 important parts of it

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…

- I feel like when the code is explained to me
  - (A) I can see the parts of it and sort of understand it
  - (B) I can see and understand only a small part of it
  - (C) I have no idea what I’m reading when I look at the code

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

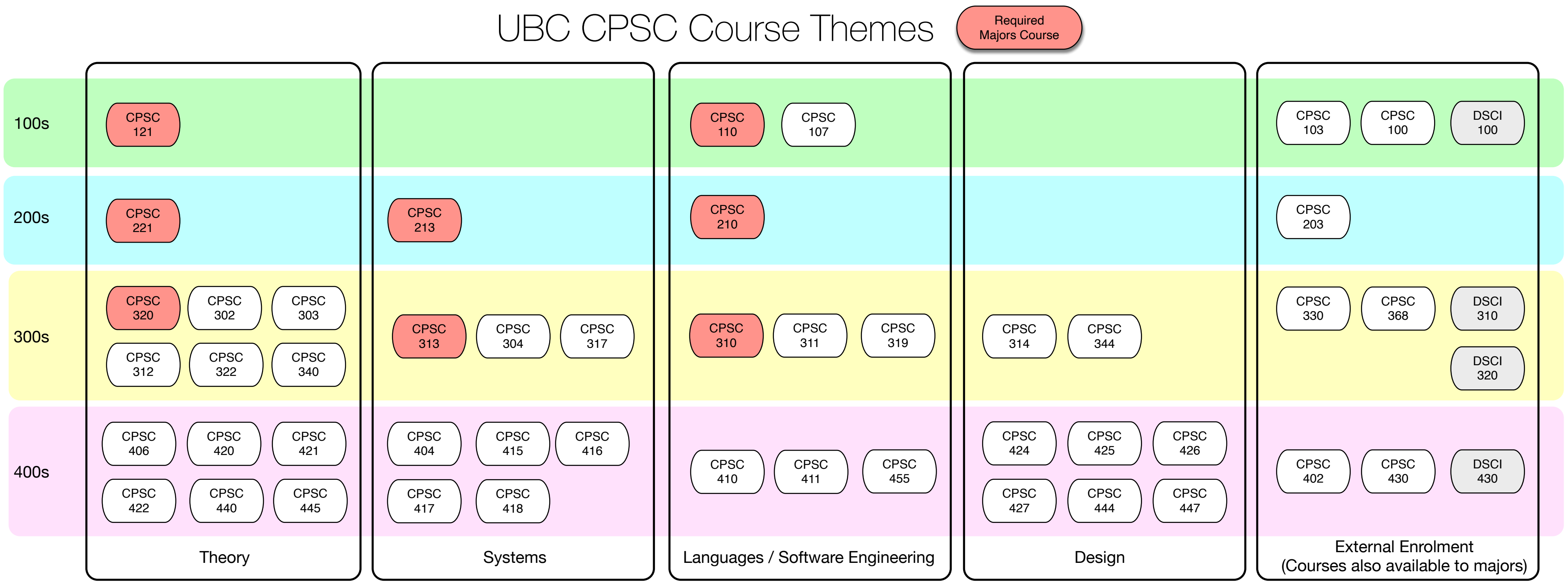
In the rest of my time at UBC

(A) I plan to take more CS courses

(B) I do not want to take another CS course

UBC CPSC Course Themes

🟥 Required Majors Course



	A	B	C	D	E
	Theory	Systems	Languages / Software Engineering	Design	External Enrolment (Courses also available to majors)
100s	🟥 CPSC 121		🟥 CPSC 110, CPSC 107		CPSC 103, CPSC 100, DSCI 100
200s	🟥 CPSC 221	🟥 CPSC 213	🟥 CPSC 210		CPSC 203
300s	🟥 CPSC 320, CPSC 302, CPSC 303, CPSC 312, CPSC 322, CPSC 340	🟥 CPSC 313, CPSC 304, CPSC 317	🟥 CPSC 310, CPSC 311, CPSC 319	CPSC 314, CPSC 344	CPSC 330, CPSC 368, DSCI 310, DSCI 320
400s	CPSC 406, CPSC 420, CPSC 421, CPSC 422, CPSC 440, CPSC 445	CPSC 404, CPSC 415, CPSC 416, CPSC 417, CPSC 418	CPSC 410, CPSC 411, CPSC 455	CPSC 424, CPSC 425, CPSC 426, CPSC 427, CPSC 444, CPSC 447	CPSC 402, CPSC 430, DSCI 430

Which kind of course most interests you now?

Faculty In A Large Research University

	exceptional subject matter expertise	exceptional teaching science and practice expertise
tenured	• Professor • Associate Professor	• Professor of Teaching • Associate Professor of Teaching
tenure-track	• Assistant Professor	• Assistant Professor of Teaching
not tenure-track		• Lecturer • Sessional

Job Search Process Points

Jobs outside universities…

- Build your own portfolio[1], 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

1. I wish this wasn't good advice, it's blatantly discriminatory.

Two summer interns

- When faced with an underspecified task at work:
  - Intern A usually goes to their manager to seek clarification before proceeding.
  - Intern B usually tries to make a reasonable decision about what is needed and proceeds.
- At the end of the summer, which one gets a permanent job offer?

At work - FtR

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

- Managers don’t have time for employees who need to be given a detailed description of what to do
  - figure it out
  - do something reasonable
  - ask only very well developed questions

- It’s easy to be fun to be around when you know what you are doing
  - teams want people who are fun to be around when they don’t know what they are doing
  - interview processes are in part trying to uncover that, they want to push you into uncomfortable territory to see how you handle that – they aren’t trying to intimidate or be macho (many of them aren’t anyways)

What About Generative AI?

- All kinds of work is about to change – maybe a lot
- Long term isn’t clear (issues go way beyond technology)
- It does seem clear that in 2-3 years or less
  - software engineers (including interns) will need to be able to use tools like Copilot (or similar) to be more <u>productive</u> (produce more value per salary dollar)
    - produce/revise code faster
    - produce better code
    - …
  - One part of that will be quickly evaluating code an AI tool proposes
  - One part of that will be having a dialog with an AI tool about code you have written
- We have tried to given you a foundation for that
  - by focusing on design level constructs that exist across all languages and tools

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?

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?