LEC 19

# CSE 121

# Final Exam Review & Victory Lap

Questions during Class?

Raise hand or send here

**sli.do #cse121**

## BEFORE WE START

***Talk to your neighbours:***

*What will you miss from winter?*
*What are you excited to leave behind?*

Music: [121 25wi lecture playlist ❄]

| | | | | |
|---|---|---|---|---|
| **Instructor:** | Matt Wang | | | |
| **TAs:** | Ailsa | Alice | Chloë | Christopher |
| | Ethan | Hanna | Hannah | Hibbah |
| | Janvi | Judy | Julia | Kelsey |
| | Lucas | Luke | Maitreyi | Merav |
| | Ruslana | Samrutha | Sam | Shayna |
| | Sushma | Vivian | | |

# Announcements, Reminders (general)

- R7 (+ extra resub) due **Thu, Mar 20th** – all assignments eligible
  - note: all extra resub emails have gone out at this point!
- today is the last day for IPL & instructor office hours
- Final Exam: **Tuesday, March 18th from 12:30 - 2:20 in KNE 130, 220***
  - look at seating charts and let me know ASAP if you're not there!
  - review Exam page of website (with policies & resources)
- TA-led review session: **Sun, March 16th from 5:30-7:50 PM, ARC 147**

# Evaluations and Awards

Please give us feedback!

- Course Evals are due **Sunday, March 16th at 11:59 PM**
  - A section eval link
  - B section eval link
- TA Evals are *also* due **Sunday, March 16th at 11:59 PM**

Bob Bandes TA award nominations open! (nominate your goat)

# Exam Review Preamble

Acknowledging: live exam review can be stressful (and feel rushed)!

Today’s goal:

- practicing how to trace through complicated code, on paper
    - helpful for tracing & comprehension problems
    - helpful for debugging problems (to find the bug)
    - helpful for programming problems (to “test” your code)
- *not* writing code or “fixing” the bug (though I’ll do that too)

# Your Task

I’ll give you about 5 minutes with a debugging spec.

Your goal is *just* to trace through the buggy code (on the back):
*why* does it produce the incorrect output?

- Assume that the random numbers from the `nextInt` calls really are 2, 9, 10, 8, and 1 (that’s not the bug).
- **<u>Do not write any code!</u>**

One note: this is from a much older 121 exam. This is not *exactly* the format of recent debugging problems, but helps us with this skill!

# Applications of CS

*or "What can I do with what I learned?" – outside of just* "write code":

- Detect and prevent toxicity online & recognize disinformation
- Help deaf & hard-of-hearing people identify sounds
- Develop a programming language that celebrates the world's languages
- Build battery-free robots & put them on insects (and… track murder hornets?)
- Computational knitting & carpentry
- Create an interactive atlas of millions of refugee experiences
- Fix Olympic badminton & identify cheating in chess
- and so much more!

# … including our assignments! (1/2)

- Computational Biology & Medicine (P2, P3)
  - fun fact: Matt did some DNA sequencing (P3+++) in grad school
  - in CSE: Chris Thachuk, Linda Shapiro, Sara Mostafavi, Su-In Lee
- Computational Art (C0, C1)
  - UW CSE has many unique intersections of CS + art!
  - "Cultural-Centric Computational Embroidery" (CSE + iSchool)
  - "Computational Illusion Knitting", "How to Knit Objects Weird"
  - "WasteBanned: Supporting zero waste fashion design"

# … including our assignments! (2/2)

- Games & Graphics (C1, C3)
  - at UW: many labs in CSE and iSchool’s GAMER group
  - fun fact: Foldit is a crowd-sourced game for protein folding
    - David Baker shared this year’s Nobel Prize in Chemistry, in part for this!!
- Social Computing (P1, C2)
  - at UW: Amy Zhang’s Social Futures Lab + so much of iSchool
- and many side quests (in lecture, section, PCM): accessibility (e.g. UW CREATE), weather forecasting, chatbots, and lots of math

# Closing the loop on P3 reflections – topics

Y’all listed many other applications of CS too! Some common areas:

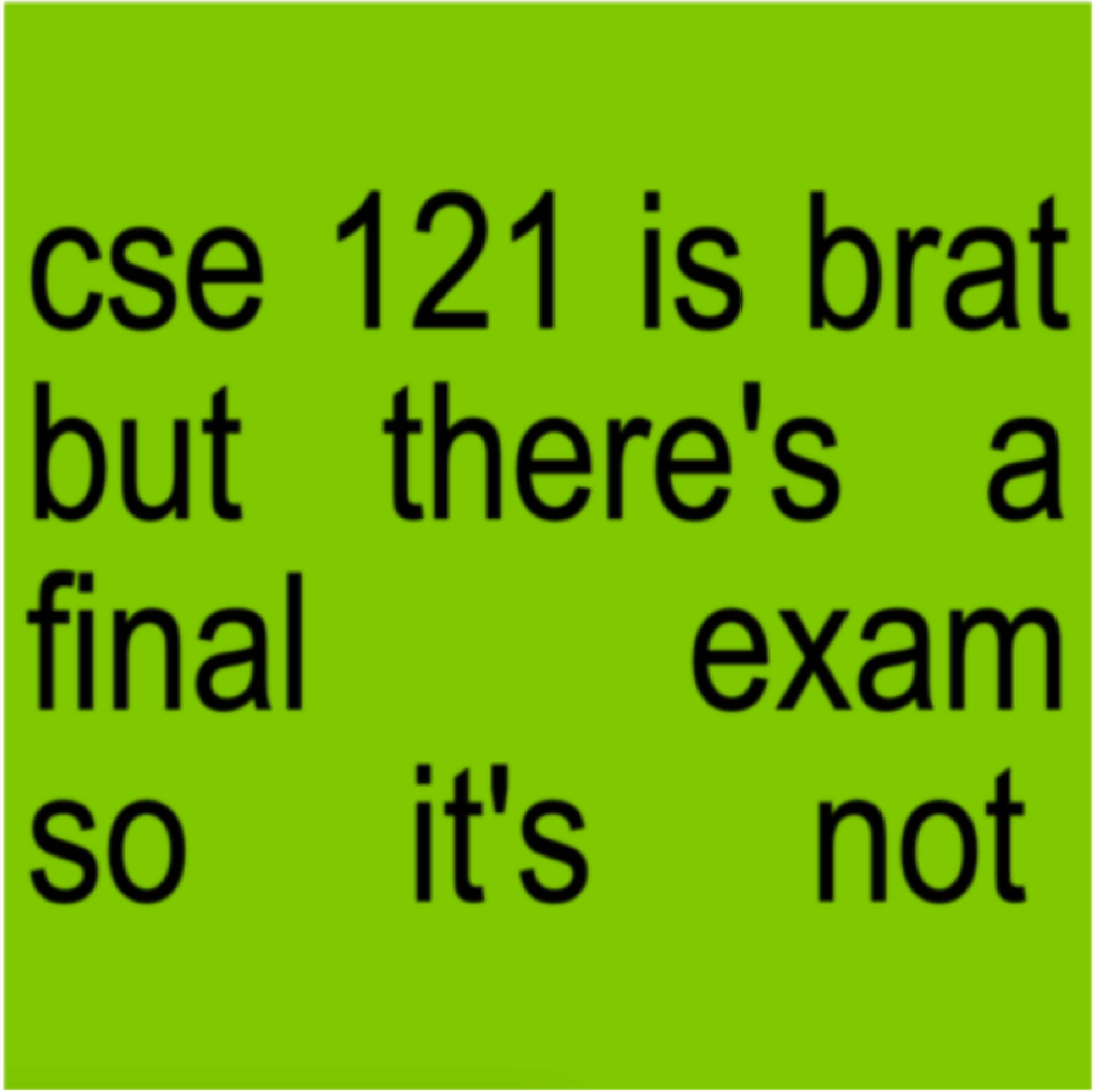
- environmental & climate science
- architecture
- physics
- sports analytics and health
- business & finance
- geography

# P3 reflections & music

“Just recently, I've used computer science in two of my interests that involve other fields.  […] The other was using it to write a wavetable synthesizer that I used to make music.”

“In music production, mixing, and sound design, you use computer programs to synthesis sounds very similar to coding, with inputs, outputs, modifications, and parameters.”

CS + music is actually *huge* and very fascinating – Matt’s first-ever internship was implementing audio processing filters. Super cool!!



cse 121 is brat but there's a final exam so it's not

# Future Courses

*or "What can I do next?"*

## Non-majors

| Course | Overview |
|---|---|
| CSE 154 | Intro. to web programming (several languages) |
| CSE 160 | Intro programming, data analysis (Python) |
| CSE 163 | Intermediate programming, data analysis (Python) |
| CSE 180 | Introduction to data science (Python) |
| CSE 373 | Data structures and algorithms (in Java) |
| CSE 374 | Low-level programming and tools (C/C++) |
| CSE 412 | Intro to Data Visualization |
| CSE 416 | Intro. to Machine Learning |
| CSE 493E | Accessibility |

## More 12X!

| Course | Overview |
|---|---|
| CSE 122 | Data structures, object-oriented programming |
| CSE 123 | More OOP, recursion |

## Majors

| Course | Overview |
|---|---|
| CSE 311 | Mathematical foundations |
| CSE 331 | Software design/implementation |
| CSE 340 | Interaction programming (mobile apps) |
| CSE 341 | Programming languages (!!) |
| CSE 351 | Hardware / Software Interface |

Other tech-related majors:
Informatics, ACMS, HCDE, Electrical & Computer Engineering, …

*See:* https://www.cs.washington.edu/academics/ugrad/current-students and https://www.cs.washington.edu/academics/ugrad/nonmajor-options/nonmajor-courses

# Closing the loop on P3 reflections – journey

We also asked you to reflect on your journey learning CS!

Impossible to adequately summarize them all, but the biggest theme was **combating myths about computer science:**

- what programming & computer science is
- what makes computer science hard (it's not just syntax!!)
- what you need to be a "good" programmer (not always math!!)
- what programmers & computer scientists look like
- **that one bad experience means you'll *always* be bad at it**

# … and some funny answers

“However, when I started hearing laufey play in the background, seeing the small ice breaker questions on the slideshow, all that dread seemed to melt away.”

“The lectures were enjoyable and I always found Matt's jokes funny, I would always giggle or chuckle at least once during lecture.”

```
String You = "CSE 121";
System.out.println("Never Gonna Give" + You + "Up");
```

# Generalizing *beyond* CS

Some of you said, ”I’m glad I took this class, but no more CS for me”

That’s <u>totally valid</u>!

Some lessons from this class that *could* apply more broadly:

- how to break big problems into smaller subproblems
- how to isolate what part of a system is broken
- attention to detail
- how to learn (and reflect) effectively

# Frequently Asked Questions

How can I get better at programming?

- Practice!

How can I learn to X?

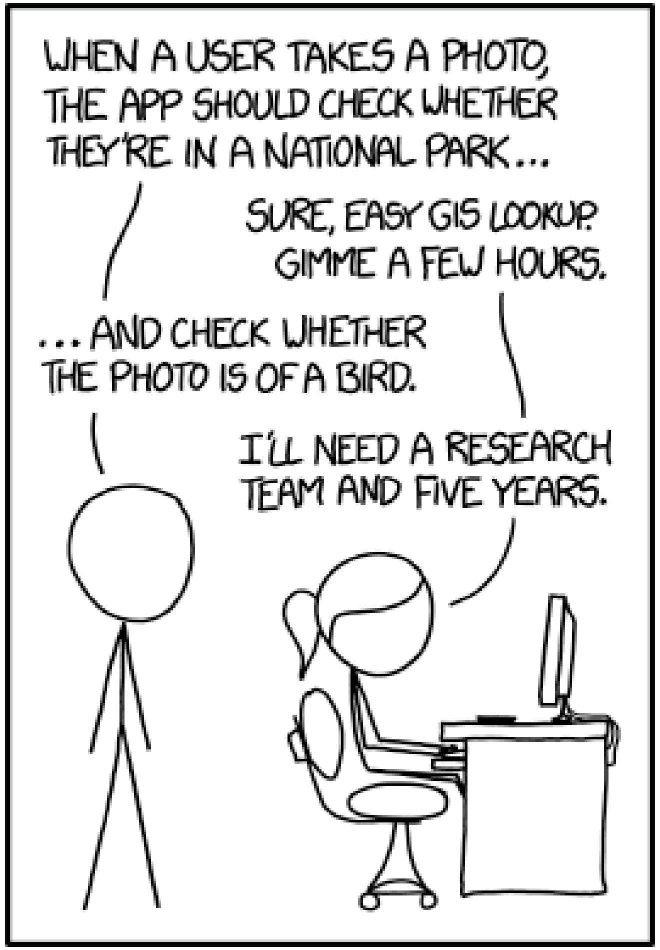
- Classes, books, videos, or self-learn!
- CS (as a field) has lots of free resources :)

What should I do next?

- Anything you're interested in!
- but: hard to tell what's easy and what's hard

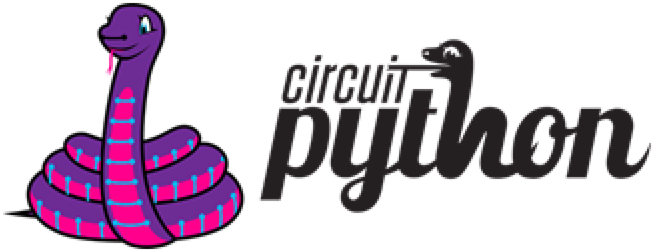
Should I learn another language? Which one?

- That depends – what do you want to do?


WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

# Aside: Cute Programming Language Logos



# Spring Break Project: Tic Tac Toe

Build your own Tic Tac Toe game + “AI”! (one of the TA’s choice ideas)

1. How would you represent a Tic Tac Toe game in Java?

   (hint: arrays will be very, very helpful!)

2. Write a method that tells you if a Tic Tac Toe game is won.
3. Write a method that gets input from the user and “makes” a move.
4. Wrap it all up – into a nice two-player game!

# Spring Break Project: Tic Tac Toe++

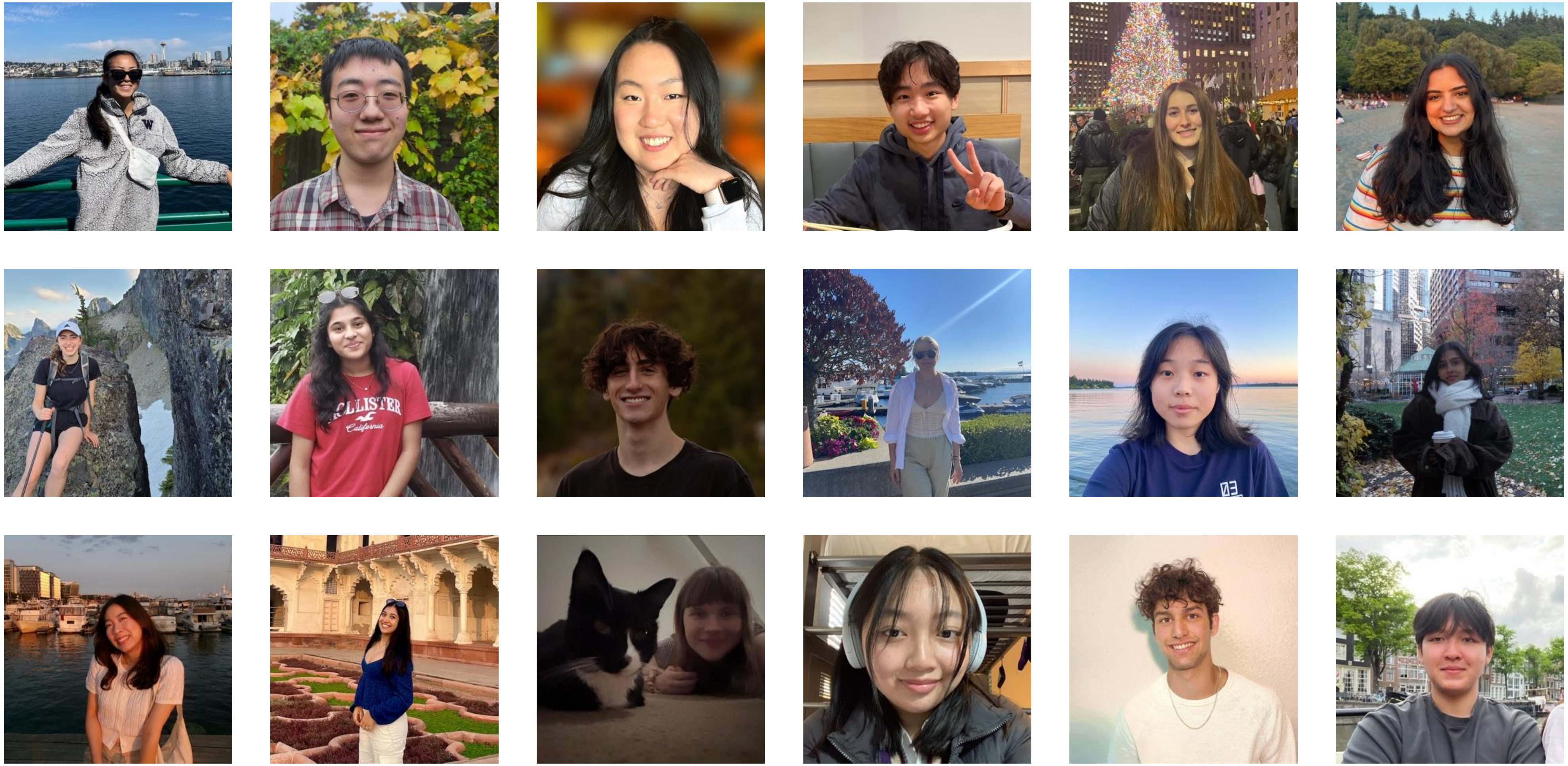
Make some “AI” that…

- just makes a random valid move (you should be able to beat this!)
- tries to make a “good” move (~ some if statements)
- never loses
  - Tic Tac Toe is a “solved game”: a perfect player will never lose.

Or, extend this idea to other grid-based games!

- similar-ish: connect four, checkers, battleship
- much harder: sudoku, chess, go, othello

# Thank your lovely TAs!



# Thank you!

Ask Me (Almost) Anything!

sli.do #cse121