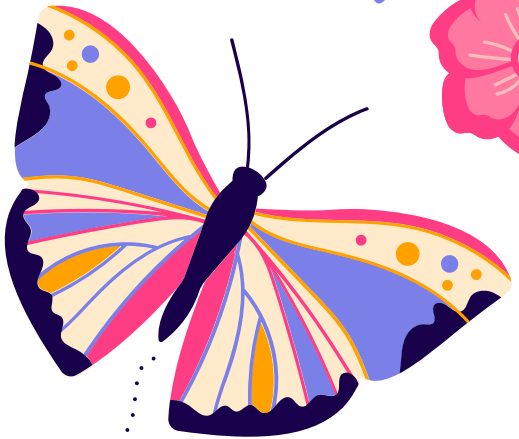
Link to the slides: https://tinyurl.com/cfuvpep6

A joint collaboration between Lucy and Noah :D

Debugging (in JS)

July 13th, 2022
CS 160 Summer 2022, Section 4A



Course reminders

Due today

Project 2: 2.9 Interactive Prototype

Due Thursday

Project 2: Final Report

Project 4: 4.1 Brainstorm

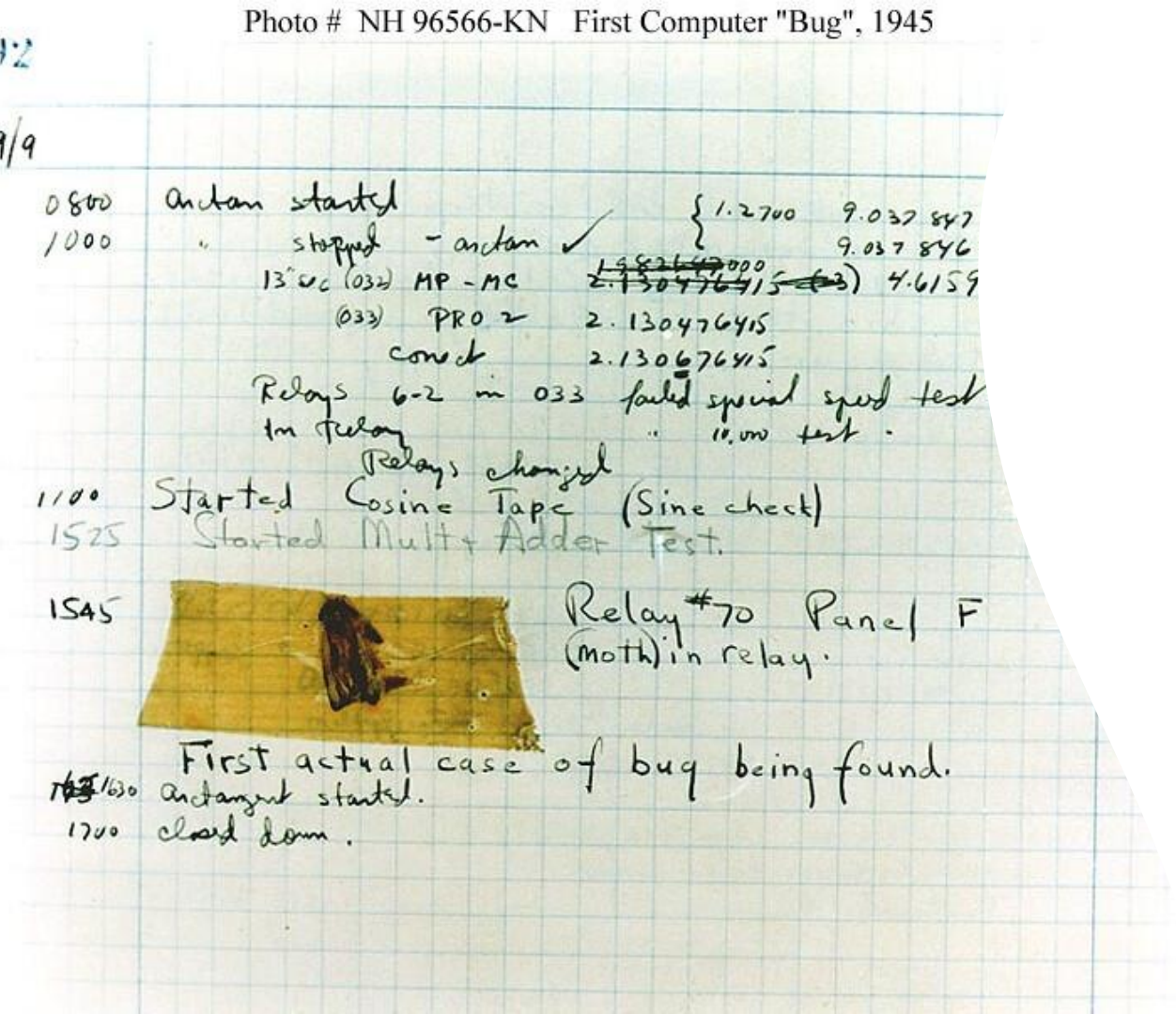
Due Friday

4.2 Interview Plan

Quiz 4

Misc.

Next week (week 5) is the last week of lectures



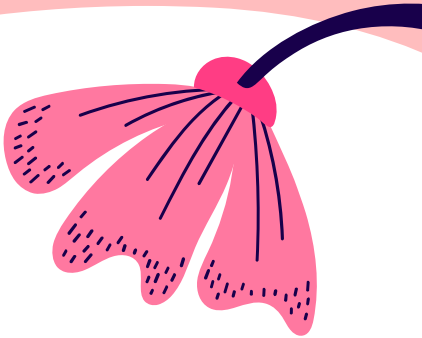
The first bug

The first ever computer bug found in Harvard Mark II computer (1945)

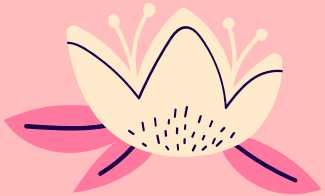
Why learn debugging?

What were some of the most frustrating experiences you had dealing with coding projects?

Why learn debugging?



For when you write 1,000 lines of code, and then your app doesn't work D:



“Anyone who has never made a mistake has never tried anything new.”

– Albert Einstein

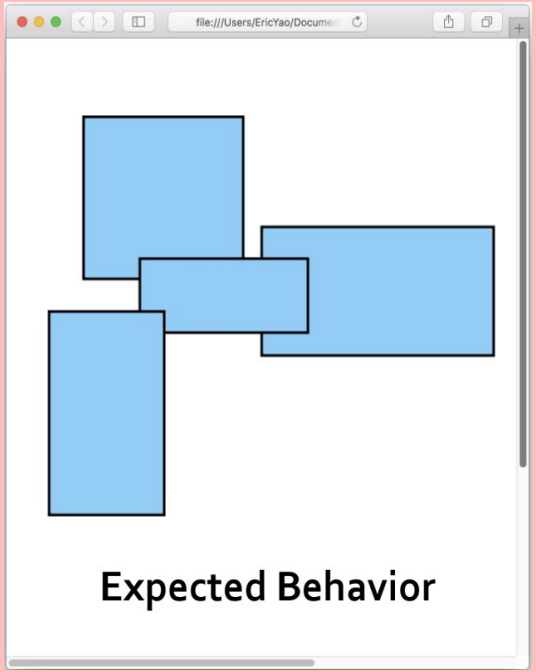
So again, why Debugging?

We know errors will happen. Might as well catch them early and minimize them!

There is a bug on both of these trees. Which one is easier to spot?

Debug the program

- A simple Paper.js program to draw rectangles (click image below)



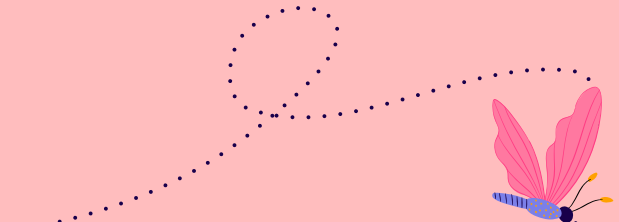
- Download the buggy and working versions

Step 1: Identify the bug

- What should the program do?
  - *When users click and drag the mouse in browser (inputs), a rectangle should appear and adjust its size based on mouse position (outputs).*
- What is the program actually doing?
  - *Nothing appears when users click and drag the mouse in the browser.*
- Is there really a bug?
  - *Yes.*

Step 2: Replicate the bug

- Find precisely specified steps to reproduce the bug
- Without them, it is impossible to verify if the bug is fixed
- Sometimes difficult
  - Asynchronous programs
  - Multi-threaded programs
  - Interactive programs



Step 3: Divide program into smaller tasks

- Debugging a large, complex program is very difficult
- Debugging each smaller, simpler task in turn is much easier

- What are the smaller tasks for the rectangle-drawing program?

Step 3: Divide program into smaller tasks

- Debugging a large, complex program is very difficult
- Debugging each smaller, simpler task in turn is much easier

- What are the smaller tasks for the rectangle-drawing program?
  - Draw a rectangle in browser without involving mouse
  - Make sure mouse event listeners work (e.g., onMouseMove())

Step 3: Divide program into smaller tasks

- Debugging a large, complex program is very difficult
- Debugging each smaller, simpler task in turn is much easier

- What are the smaller tasks for the rectangle-drawing program?
  - Draw a rectangle in browser without involving mouse
  - Make sure mouse event listeners work (e.g., onMouseMove())
  - Send and receive a line (with two hard-coded endpoints)

Step 3: Divide program into smaller tasks

- Debugging a large, complex program is very difficult
- Debugging each smaller, simpler task in turn is much easier

- What are the smaller tasks for the rectangle-drawing program?
  - Draw a rectangle in browser without involving mouse
  - Make sure mouse event listeners work (e.g., onMouseMove())
  - Send and receive a line (with two hard-coded endpoints)
  - Finally, send and receive a line as it is drawn in real-time

Step 4: Debug each task in turn

- Step 4.1: Further divide into sub-tasks, if possible
- Step 4.2: What should each sub-task do?
- Step 4.3: Use tools below to debug each sub-task:
  - ✨✨***Google***✨✨
  - Comment out irrelevant code and maybe add code snippets
  - Check error messages in JavaScript console
  - Use console.log() to print
  - Use breakpoints to step through code (view an example here)
  - Hypothesize the bug, design and run tests, and fix the bug

Step 4: Debug each task in turn

- Step 4.1: Further divide into sub-tasks, if possible
- Step 4.2: What should each sub-task do?
- Step 4.3: Use tools below to debug each sub-task:
  - ✨✨***Google***✨✨
  - Comment out irrelevant code and maybe add code snippets
  - Check error messages in JavaScript console
  - Use console.log() to print
  - Use breakpoints to step through code (view an example here)
  - Hypothesize the bug, design and run tests, and fix the bug

Tips for debugging

- Add comments to your code as you're writing it!
  - You'll thank yourself later
- Google your error messages (or just read them in general)
- Talk to a code duck
- Have no shame and use print statements
- Write Unit-Tests
- Brainstorm all edge and corner cases

Activity

- Download the exercise here (see expected behaviors here)

- Step 1 & 2. Identity and replicate the bug
- Step 3. Divide the program into smaller tasks
- Step 4. Debug each task in turn, using
    - ✨✨*Google*✨✨
    - Comment out or add code snippets
    - Error messages, console.log(), and breakpoints in console
    - Hypothesize the bug, design and run tests, and fix the bug

- Download the solution here

Thank you for coming!

Do you have any questions?
Lucy's OH - Fridays at 3 PM
Noah's OH - Thursdays at 3 PM

The slides were taken from Eric Yao's CS160 presentation from a previous semester