ECS 261: Program Verification

Lecture 0 – SQ2025

Welcome!

- This is a **graduate course** in a topic called **program verification**
  - AKA **formal verification**, more on this soon

- Current cap: 50
  - Some will drop – more expected off waitlist

Your instructor

Your Instructor: Prof. Stanford (Caleb is also OK)

Started at Davis: July 2023

DavisPL Research Group

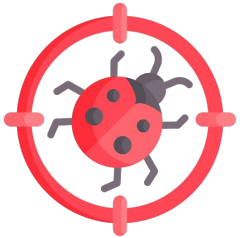
Website

Plan for today

1. What is this class about?
2. FAQ, syllabus, and logistics
3. Demos (time permitting)

What is this class about?

What is this class about?



We live in a world of buggy software

Crashes… Crowdstrike (2024)

CYBERSECURITY

$1.94B in Expected Healthcare Losses Due to CrowdStrike Disruption

CrowdStrike released a root cause analysis of the incident that caused a global outage on July 19

Pietje Kobus

Aug. 13, 2024

CROWDSTRIKE

Crashes… Crowdstrike (2024)

CYBERSECURITY

$1.94B in Expected Healthcare Losses Due to CrowdStrike Disruption

8.5 million devices were confirmed affected by the CrowdStrike outage, but Microsoft says that's only a subset.

CROWDSTRIKE

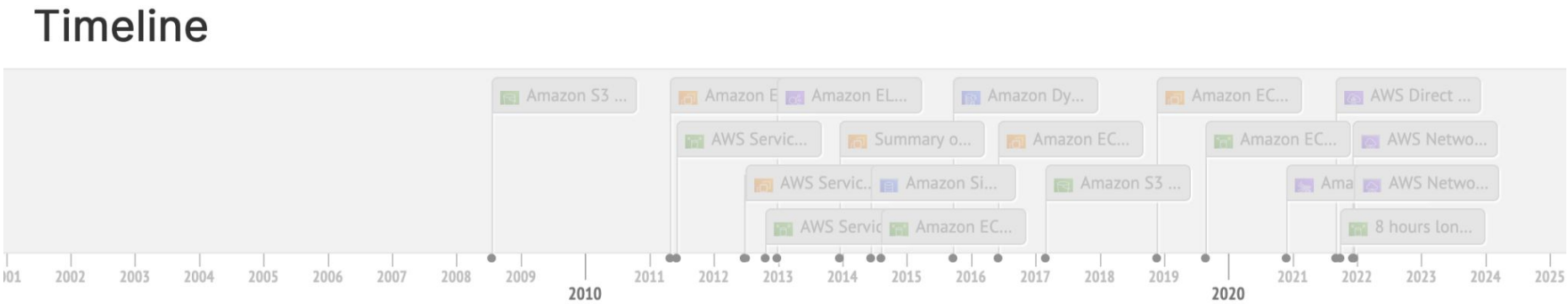
Crashes… Crowdstrike (2024)

Crowdstrike → device driver → operated in kernel mode (full admin access)

- Crowdstrike driver certified by Windows (WHQL) as being compatible!
- However, driver was written to **load definition files** at runtime – “Agile development” “customers should get the latest protection” – see: Zero day attacks
- Invalid definition file (just contained 0s) - accessed an invalid pointer (0x9c)
  - Definitions can execute arbitrary code and no longer certified!
- Driver marked as a “**boot start driver”** – couldn’t be turned off!

Full explanation here: https://www.youtube.com/watch?v=wAzEJxOo1ts

Outages… AWS (2010 to present)



Source: https://awsmaniac.com/aws-outages/

Outages… AWS (2010 to present)

Timeline

OUTAGES SECURITY & RISK MANAGEMENT

AWS Outage that Broke the Internet Caused by Mistyped Command

Amazon says Tuesday's mayhem resulted from mistake during a routine debugging exercise.

Source: https://awsmaniac.com/aws-outages/

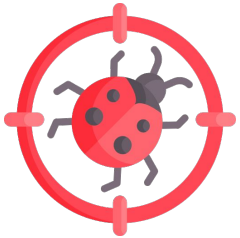
Security bugs…

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

Lost data…

hoakley / March 18, 2024 / Macs, Technology

Serious bug in Sonoma 14.4 will destroy saved versions in iCloud Drive



We live in a world of buggy software

You might have even encountered your own bugs “in the wild”...

File syncing service (Oct 2018)

On Mon, Oct 22, 2018 at 02:33 AM, "Caleb Stanford" <caleb_stanford@alumni.brown.edu> wrote:

•••

> Hello,
>
> I would like to report a bug in which Insync deleted 2 of my files while running solely in the background. I have included information below. After a progressive panic attack trying to find the files, I was finally able to find them in the .insync-trash folder after looking at this page https://help.insynchq.com/resolving-and-reporting-issues/general/retrieving-deleted-or-missing-files.

File syncing service (Oct 2018)

On Mon, Oct 22, 2018 at 02:33 AM, "Caleb Stanford" <caleb_stanford@alumni.brown.edu> wrote:

•••

Hello,

Steps to reproduce:

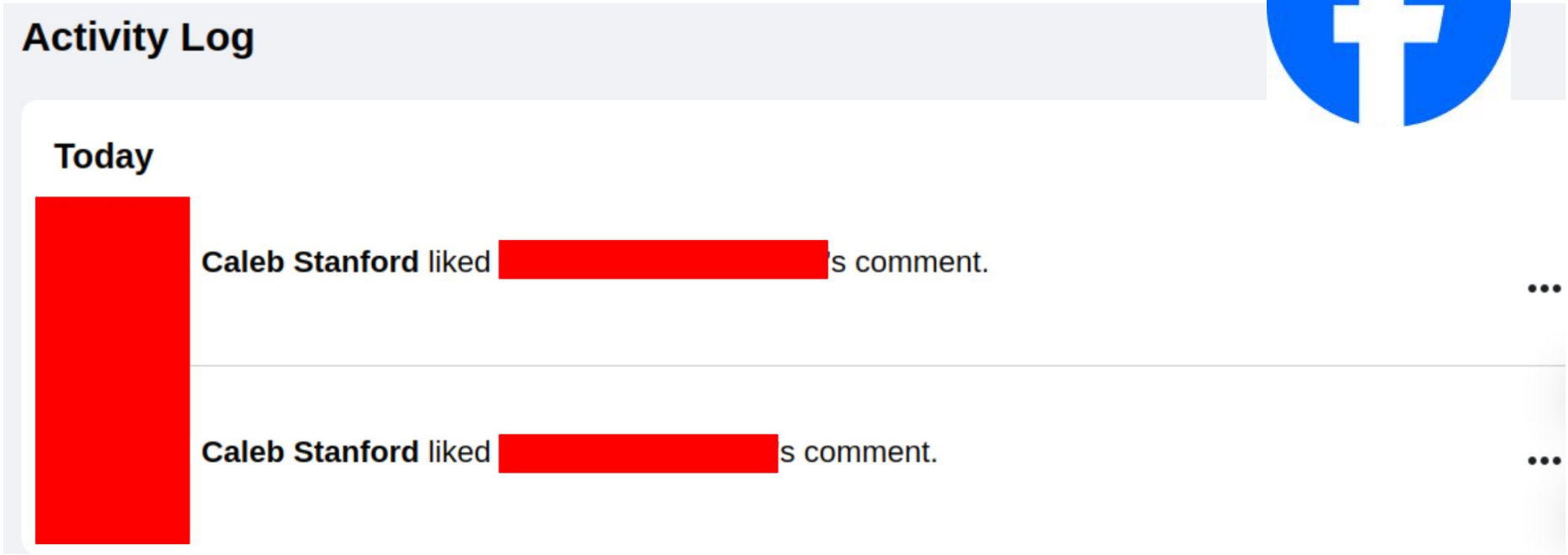
1. Start with folder A with many files and a folder B somewhere else

2. Move the files from folder A to folder B by a single cut-and-paste operation

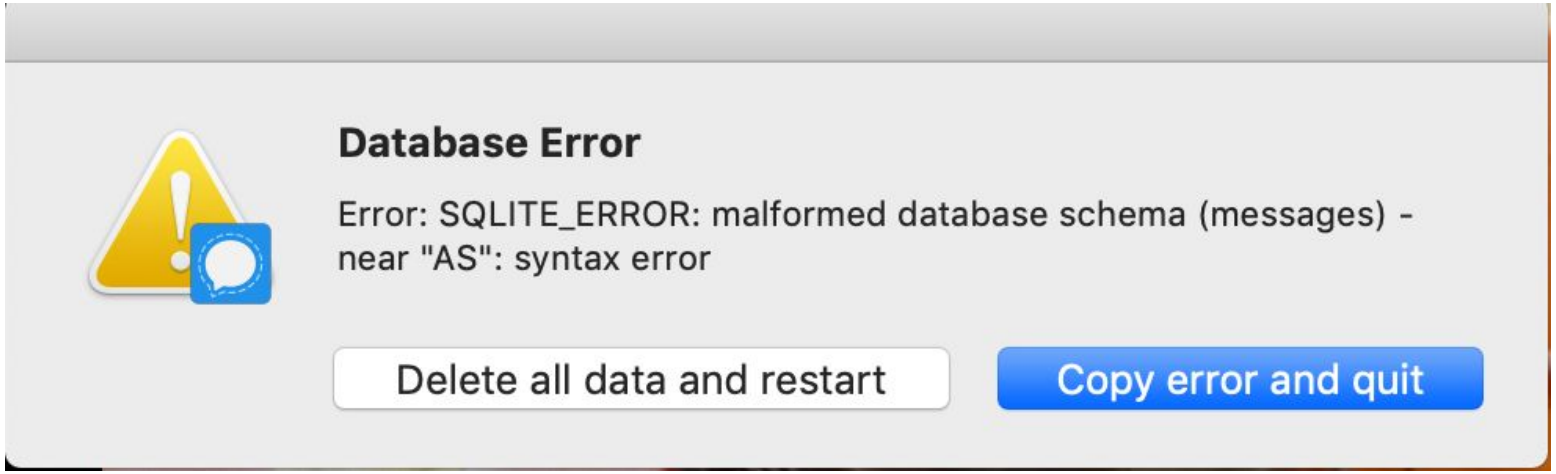
3. While Insync is still processing this move, delete folder A

4. Insync may delete files from folder B, after they are already moved by the system, if the file no longer exists in folder A. The files will be moved to .../.insync-trash/.../B and can be recovered from there.

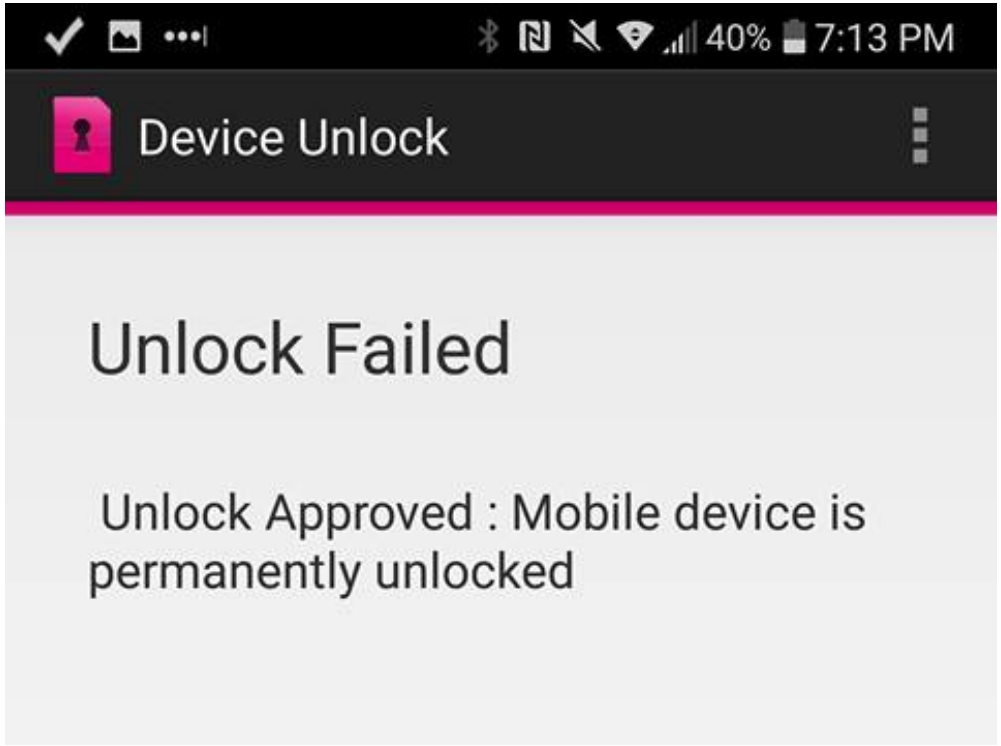
Social media (May 2021)



Signal messaging app (March 2022)



⁉ (Unknown from November 2018)



🙃 (Unknown from January 2019)

Your example here? (optional)

By show of hands

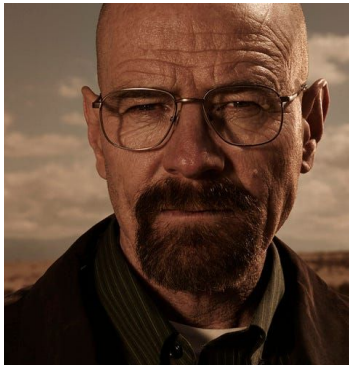
Making the situation worse…

The bug may only show up on **some platforms**

It may require an **esoteric/obscure** input 

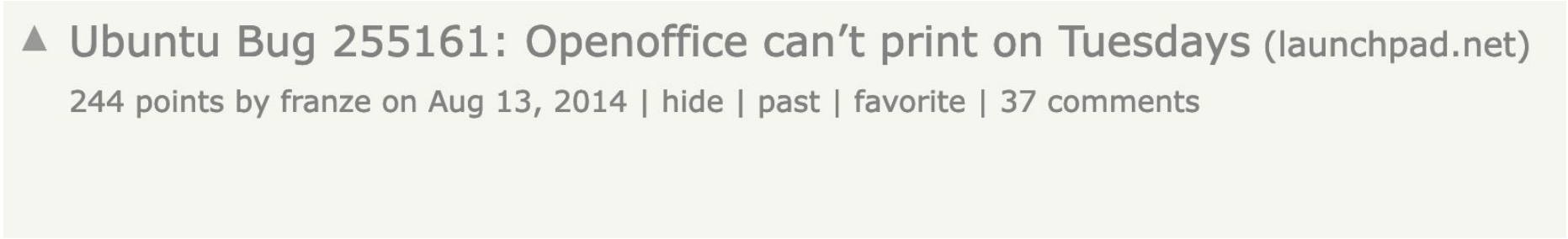
Or fail to show up at all.

Heisenbug (n.): a software bug that seems to disappear or alter its behavior when one attempts to study it.



Heisenbugs

Heisenbug (n.): a software bug that seems to disappear or alter its behavior when one attempts to study it.



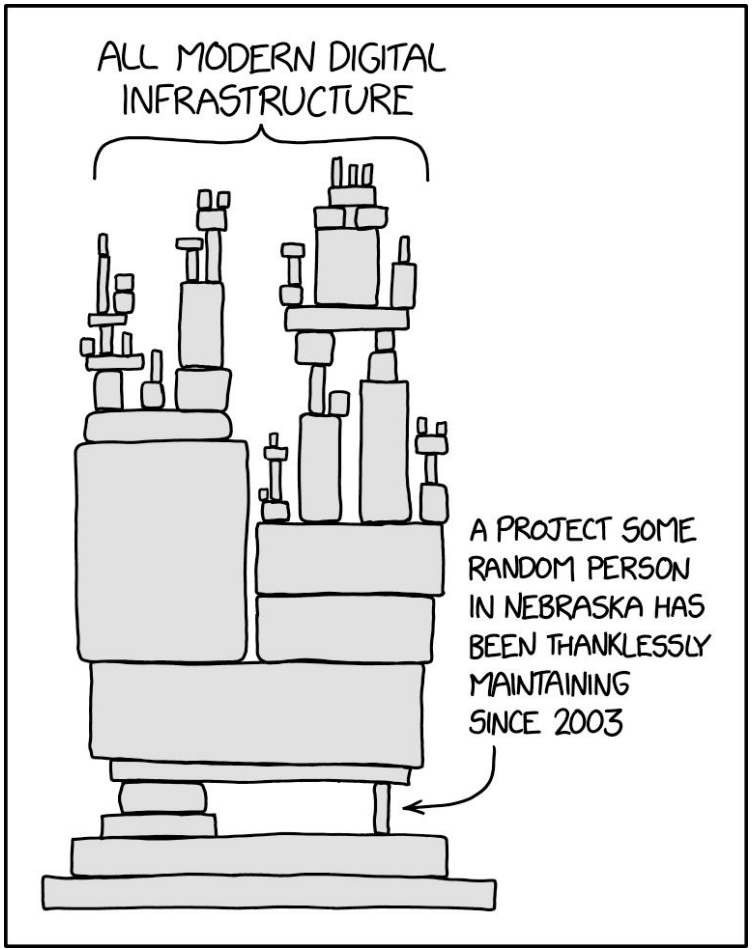
We live in a world of buggy software

Why is software like this?

Why is software like this?

Are we all just terrible programmers?

Why is software like this?



Credit: xkcd

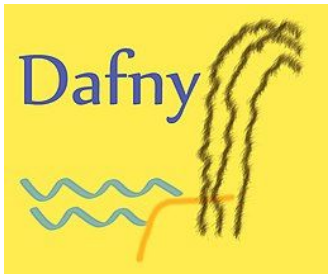
Is there a better way?

Is there a better way?

1. Write down what the program does (and should do)
2. Come up with a rigorous mathematical argument
3. Use automatic tools check (verify) that argument

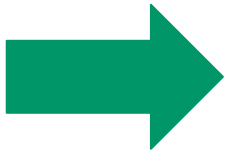
Is there a better way?

1. Write down what the program does (and should do)
2. Come up with a rigorous mathematical argument
3. Use automatic tools check (verify) that argument



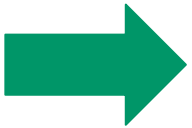
Is there a better way?

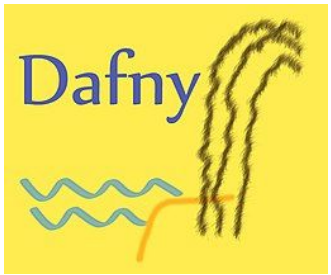
What this class is about



1. Write down what the program does (and should do)
2. Come up with a rigorous mathematical argument
3. Use automatic tools check (verify) that argument

Real-world tools used in industry





Discussion question

1. Describe what the program **does**
2. Come up with a definition for what this program **should do**
3. Share your definition with your neighbors

```
def is_even(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    elif x == 2:
        return True
    elif x == 3:
        return False
    elif x == 4:
        return True
    else:
        return False
```

Then fill out this poll:

https://forms.gle/dhXuzDsopEMiirHt8

Sharing your answers

1. Describe what the program **does**
2. Come up with a definition for what this program **should do**
3. Share your definition with your neighbors

```
def is_even(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    elif x == 2:
        return True
    elif x == 3:
        return False
    elif x == 4:
        return True
    else:
        return False
```

Then fill out this poll:

https://forms.gle/dhXuzDsopEMiirHt8

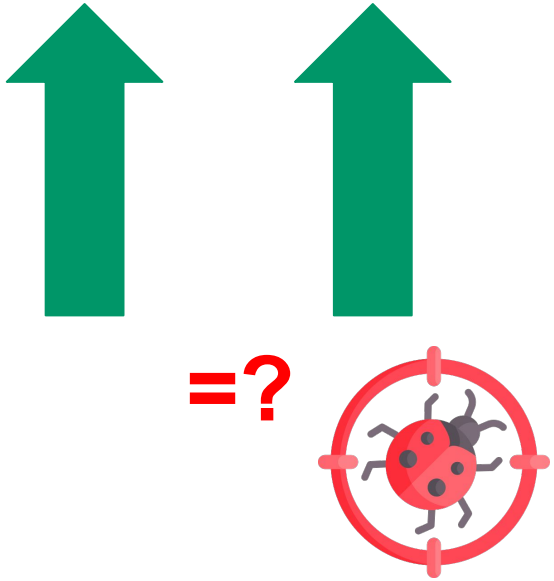
The underlying question: **What is a bug?**

The underlying question: **What is a bug?**

Answer: It Depends!

(Remember that first step)

1. Write down what the program does (and should do)



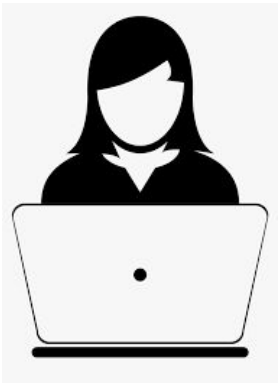
Fundamental question in software development!

Example (optional or skip for time)

Please build me a car

Example

Please build me a car

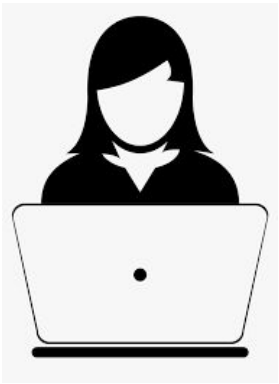


What car? What is the definition you have in mind?

Example

Please build me a car

It should have four wheels and you can drive it places



What car? What is the definition you have in mind?

Example

Please build me a car

It should have four wheels and you can drive it places

Ok, here you go

Example

Please build me a car

… it should also have a roof and seats inside

Ok, here you go

Example

Please build me a car

… it should also have a roof and seats inside

Ok, here you go

Example 2

Please write a program to check if a number is even or odd

```python
def is_even(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    elif x == 2:
        return True
    elif x == 3:
        return False
    elif x == 4:
        return True
    else:
        return False
```

Ok, here you go

Example 2

Please write a program to check if a number is even or odd

```
def is_even(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    elif x == 2:
        return True
    elif x == 3:
        return False
    elif x == 4:
        return True
    else:
        return False
```

Ok, here you go

Is this correct?

(Why not?)

Example 2

Please write a program to <do some task>

```
def is_even(x):
    if x == 0:
        retur
    elif x ==
        retur
    elif x ==
        retur
```

Ok, here you go

Is this correct?

(Why not?)

I apologize for the mistake in my previous response. You are correct that the output I provided was not correct. Thank you for bringing it to my attention.

Here is the corrected code to standardize the data:

python Copy code

```
import numpy as np
```

Example 2

Please write a program to <do some task>

```
def is_even(x):
    if x == 0:
        retur
    elif x ==
        retur
    elif x ==
        retur
```

Ok, here you go

Is this correct?

(Why not?)

The Register
https://www.theregister.com › chatgpt_stack_overflow_ai

ChatGPT gets code questions wrong 52% of the time

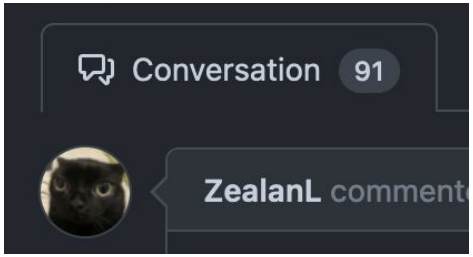
Aug 7, 2023 — **ChatGPT**, OpenAI's fabulating chatbot, produces **wrong** answers to **software programming** questions more than half the time, according to a study ...

Example 3

Please write a program to play chess

Ok, here you go

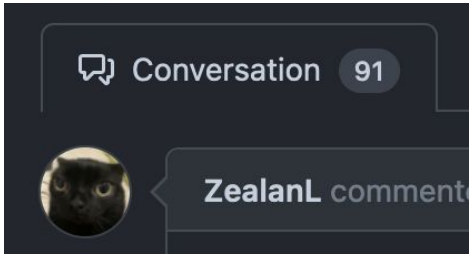
Example 3



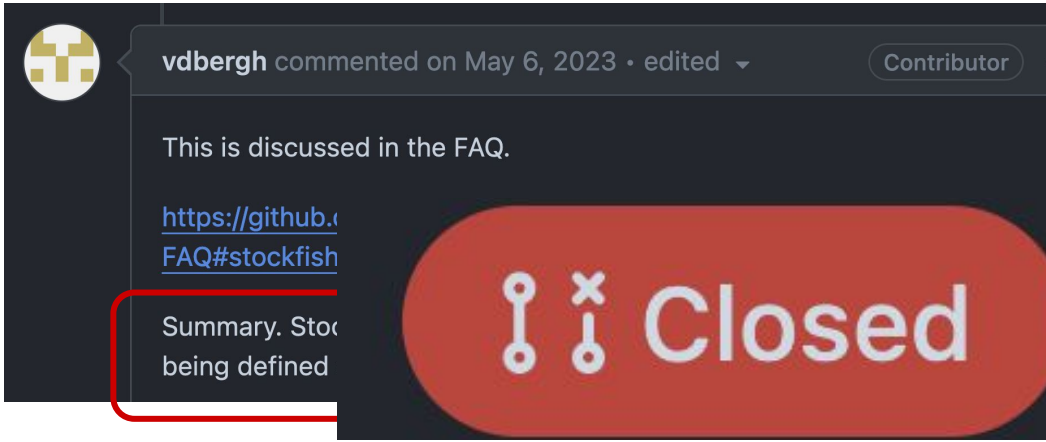
Your program can be used by a bad actor to access and modify arbitrary user memory?

It’s a feature, not a bug

Example 3



Your program can be used by a bad actor to access and modify arbitrary user memory?



It’s a feature, not a bug

The problem

We need a

clear and unambiguous

way to determine if programs are correct.

The problem

We need a

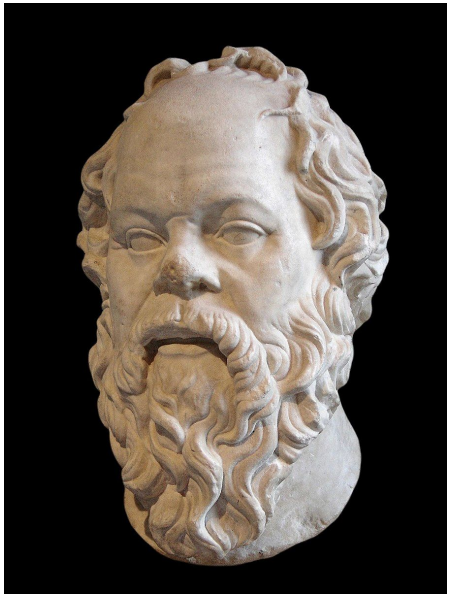
clear and unambiguous

Everyone should agree!

way to determine if programs are correct.

That is:
What the software is;
What it is supposed to do; and
Why it works (or why it doesn't)

Clear and unambiguous?



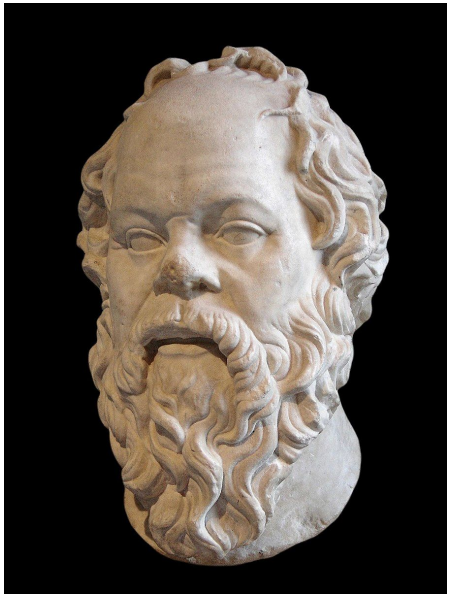
All men are mortal.

Socrates is a man.

Therefore, Socrates is mortal.[2]

Logical Syllogism

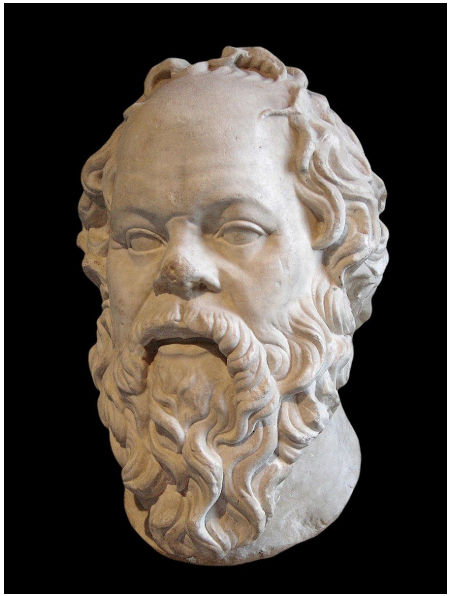
Clear and unambiguous?



A *proof* is a rigorous mathematical argument that demonstrates why a given answer is correct

even to the most serious skeptic.

Clear and unambiguous?



$*54{\cdot}43.\quad \vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

$[*51{\cdot}231] \qquad \equiv . \iota`x \cap \iota`y = \Lambda .$

$[*13{\cdot}12] \qquad \equiv . \alpha \cap \beta = \Lambda \qquad (1)$

$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash .$ Prop

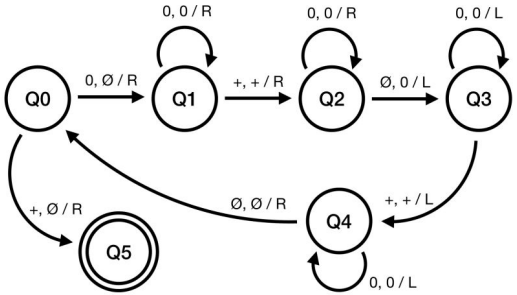
From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

(Don’t worry, I won’t ask you to write this)

Everything is Logic

Proofs can be applied to programs!

Programs are mathematical objects



[0, 1, 2, 3, …]

We can test if the program is correct by (again):

1. Writing down what the program does (and should do)
2. Coming up with a rigorous mathematical argument
3. Using automatic tools check (verify) that argument

⬆️ Program verification in a nutshell

Tools used in industry

Testing tools

(Many others)

Tools used in industry

Automated theorem provers

Microsoft Research

Z3

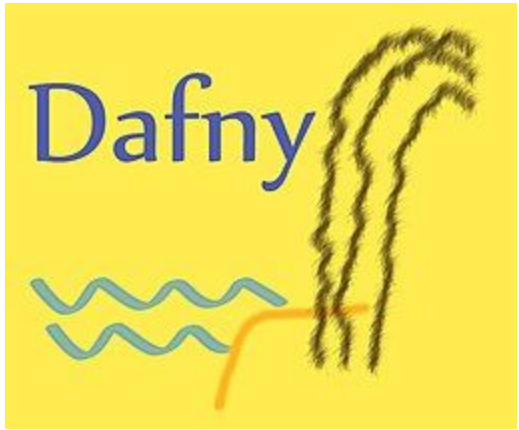
Z3 is a theorem prover from Microsoft Research. It is licensed under the MIT license.

aws Zelkova

"The total number of invocations of Zelkova ranges from a few million to tens of millions in a single day"

Tools used in industry

Program verifiers



Idris

Agda

(Focus of this class)

Recap: what this class is about

1. Write down what the program does (and should do)
2. Come up with a rigorous mathematical argument
3. Use automatic tools check (verify) that argument

Some related goals… (borrowed from ECS 189C)

Concepts and **tools** that help you understand:

1. What the software is;
2. What it is supposed to do; and
3. Why it works (or why it doesn't)

Plan for today

1. What is this class about?
2. FAQ, syllabus, and logistics
3. Demos (time permitting)

FAQ

A general disclaimer

To my knowledge, this class has not been taught **since 1997**

A general disclaimer

To my knowledge, this class has not been taught **since 1997**

- Topics listed in the catalog are out of date; please view the course syllabus for up-to-date information
- Some aspects of the course will be experimental; please expect some content and plans to be in flux!

Q: Are there any prerequisites?

A: No formal prerequisites

- Some familiarity with mathematical proof (e.g. ECS 20/120) and mathematical logic (e.g. Phil 112) is helpful, but not required
- I will assume a basic programming background
  - (e.g., ability to write FizzBuzz, ECS 36A/36B/36C)
- Encourage a **hands-on** approach: the tools can help you check your work

Q: Will I be required to write mathematical proofs?

A: Yes (a little bit)

- Writing mathematical proofs yourself in important part of program verification!
- BUT: encourage a **hands-on** approach: the tools we will cover can help you check your work

Q: Can I take this class as an undergraduate?

A: Yes

- Course assumes no undergraduate background in formal verification tools, so is appropriate for undergraduates or graduates

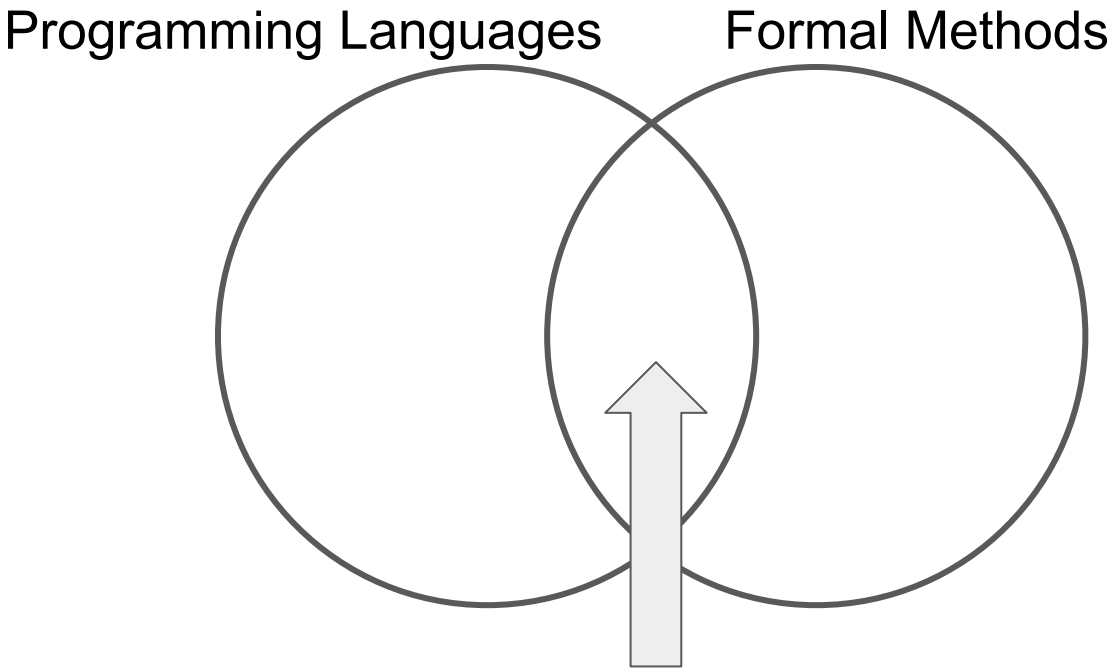
Q: Can I take this class if I've already taken ECS 189C (Spring 2024)?

A: Yes, but…

- As this course assumes no background, there will be a substantial overlap with 189C
- Some of the homeworks will also overlap
- The final project will be new and if you choose to take this option, I will encourage you to go above and beyond on the final project

(I won't stop you)

Q: What area of computer science is ECS 261?



Related areas: SE, security, theoretical computer science, …

Q: Does this course count as a graduation requirement?

A: **Yes,** ECS 261 counts towards your graduation if you are using the **new graduation requirements** approved this year

- **Software bucket** – 3 buckets, 4 units in each bucket

Software	ECS 231	Large-Scale Scientific Computation	4
	ECS 235A	Computer and Information Security	4
	ECS 245	Analysis of Software Artifacts	4
	ECS 260	Software Engineering	4
	ECS 261	Program Verification	4

Q: Is this course right for me?

Short answer: Probably!

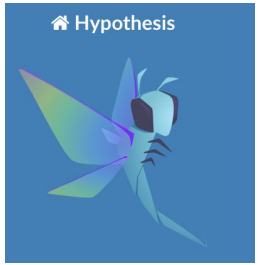
Long answer: Especially if:

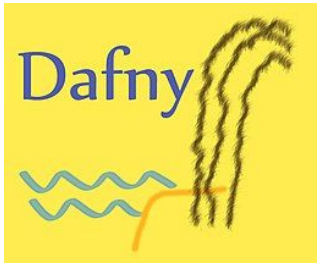
- You want to know the fundamental principles of software verification and learn about tools that are used in industry
- You are interested in thinking mathematical or logically about software and what it does

Syllabus & logistics

Learning objectives

- Understand the concept of software verification and its importance
- Understand and apply automated verification tools like Z3 for software analysis and logical reasoning tasks.
- Understand and use dedicated program verification tools such as Dafny to develop verified software.
- Understand the logical underpinnings of verification tools, and program logics for program reasoning.





Waitlist

During the first/second weeks: Please do attend the class

- Some people will drop
- Unfortunately, I can’t issue any PTAs

End of the second week: let me know if you are still on the waitlist

- Deadline to drop: 10th day of the quarter/instruction (drop deadline)
- Depending on interest, I will request to increase the enrollment cap

TL;DR: Please attend the lectures even if you are on the waitlist.

Graded work

- **Participation (10%)**
- **Homeworks (20%)**
- **Exams (30%)**
- **Final Project (40%)**

Attendance and participation (10%)

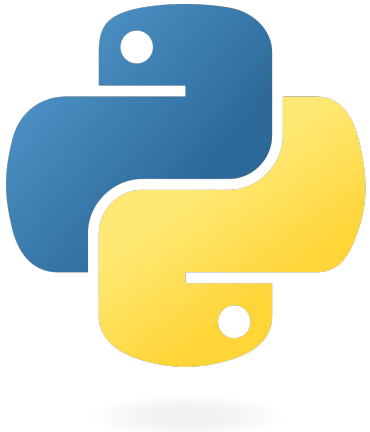
Fill out the in-class polls (participation points only)

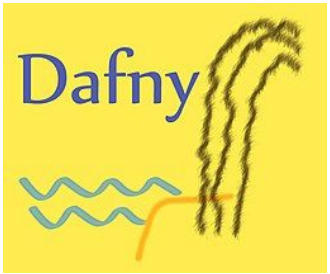
If you are sick: Starting from Wednesday, you may join the class remotely via Zoom (the quality may not be as good)

If you miss class: Lectures are recorded. You can make up the in-class polls at any time

Homeworks (20%)

About 3 homeworks are planned – using (1) Z3 in Python and (2) Dafny to help get you up to speed on working with these tools for the project





Homework 0: installation help

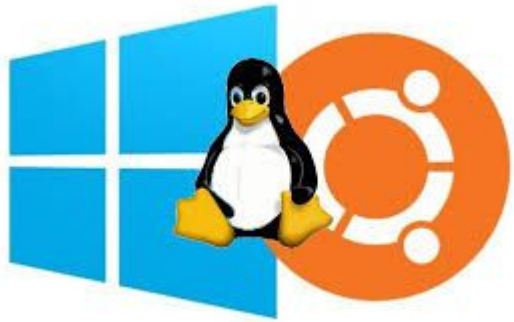
(submission link posted soon, probably Thu/Fri)

Homework 0: installation help

Please note: I recommend using **MacOS or Linux**

If you are on Windows, I recommend **Windows Subsystem for Linux (WSL)**

- Well-engineered tool and very useful to know about when doing software development work in the real world!
- https://learn.microsoft.com/en-us/windows/wsl/install



Homework (and Project) Grading

Most important: please run your code

- Software engineering is about running software!
- Running and testing your code frequently is an important part of developing software in the real world – including ensuring your code works on someone else's machine
- Please double check! We cannot give credit to code that doesn't run.

Exams (30%)

- I am planning to do a single exam, either a midterm or a final
- Why exams? (more on this in a bit)

Project (40%)

Formal verification applied to a real-world software project of your choosing!

I will announce more details in class.

Piazza

Please join and monitor Piazza!

ECS 261 on Piazza

- Don’t email me, post to Piazza!
- Make your post public and (if you prefer) anonymous
- Additional platforms: Canvas and Gradescope

AI Policy

In general: AI collaboration is allowed and encouraged, but not required

Some advice:

- I encourage you to use AI in a way that is helpful to you, and to use caution that your use of AI is aiding (and not preventing) your own understanding of the material and critical thinking skills.
- AI has trouble with Dafny and Z3. Don’t fall into the trap of assuming AI is right!
- Exam will be in-class and closed-book
- Advice from Jason Lowe-Power

Collaboration Policy

- Collaboration is allowed and encouraged!
- You can work on the homeworks and final project in groups of 2-3
- Please list your collaborators at the top of your homework
- Everyone should submit their own solution (for the homeworks); group submissions (for the project)

A Rough Schedule (subject to change)

We have about 10 weeks, here is a rough plan:

- Week 1: introduction to program specifications
- Weeks 2 and 3: introduction to logical reasoning tools, Z3
- Weeks 4, 5, 6, 7: deep dive into Hoare logic, Dafny
- Week 8: advanced topics (if time permits)
- Last 2-3 weeks: final project presentations

Tentative Schedule

Communication

TA: **Enzuo Zhu**

Office hours: TBD (will be posted on Piazza)

Please use Piazza for questions (not email)

Respect and discrimination

Please be nice!

Include everyone in group discussions

Reach out to me if there are any problems

Late policy

- Polls and HW0 can be made up
  - You don’t need to email me about this, just submit the poll! :)

- For HW, I like to be lenient in case more than one or two people are struggling! I also accept late work (though I do not guarantee that it will be graded), I encourage you to keep working on an assignment if you didn’t finish it by the deadline.
- Final project: typically can’t be extended since we need you to be ready for your presentation in time.

Other disclaimers

- UC Davis Job Scams Prevention
- Policies against harassment and discrimination
- List of resources for student health and well-being
- SISS for international student issues (and reach out to your graduate advisors)

Questions for me?

Plan for today

1. What is this class about?
2. FAQ, syllabus, and logistics
3. Demos (time permitting)

Demos

✨ Puzzle ✨

I’m thinking of 2 numbers.

The +, *, -, and / of the numbers are (not necessarily in this order):

20, 95, 105, 500

What are the numbers?

✨ Puzzle ✨

(Another one)

The +, *, -, and / of the numbers are (not necessarily in this order):

2, 6, 18, 72

What are the numbers?

✨ Puzzle ✨

(Another one)

The +, *, -, and / of the numbers are (not necessarily in this order):

2, 6, 18, 72

Is this always possible?