

# ECS 189C: Software Correctness

(Special Topics in Programming Languages & Compilers)

Lecture 0 – SQ2024

# Welcome!

- This is a **special topics course**
- [Flyer](#)
- Cap: 60 BUT some will drop – more expected off waitlist
- 4 units



**Please remember to change the number of units to 4**

# About the Instructor

What I do: Programming Languages

(Started at Davis: July 2023)

[Website](#)



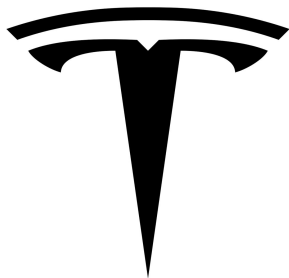
DavisPL Research Group

# Plan for today

1. What is this class about?
2. Syllabus and logistics
3. Demo

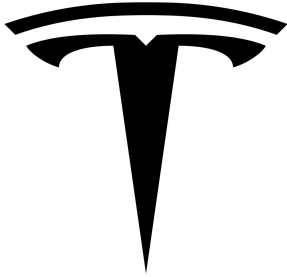
What is this class about?

# Example

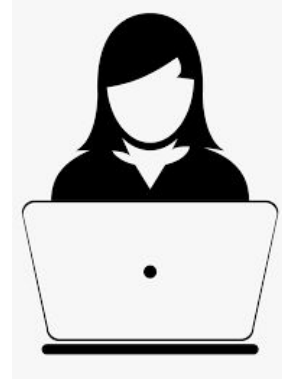


Please build me a car

# Example



Please build me a car



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

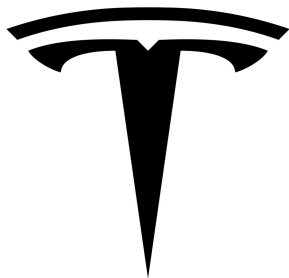
Slido

Join at  
**slido.com**  
**#2041 532**



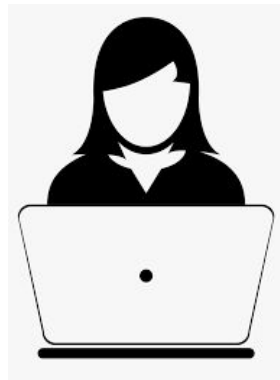


# Example



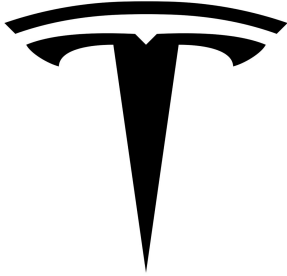
Please build me a car

It should have four  
wheels and you can  
drive it places



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

# Example



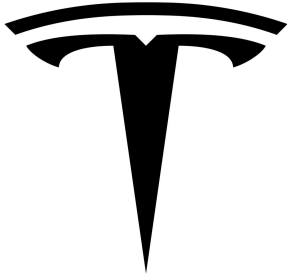
Please build me a car

It should have four  
wheels and you can  
drive it places



Ok, here you go

# Example



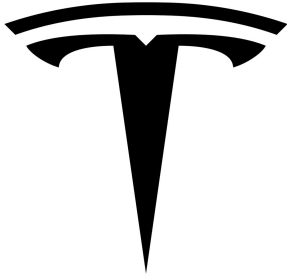
Please build me a car

... 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

# What does this have to do with programming?



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

# What does this have to do with programming?



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
```



Is this correct?  
(Why not?)

Ok, here you go

# What does this have to do with programming?

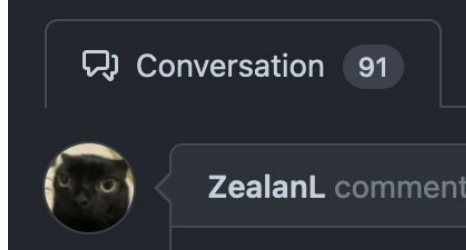


Please write a program  
to play chess



Ok, here you go

# What does this have to do with programming?



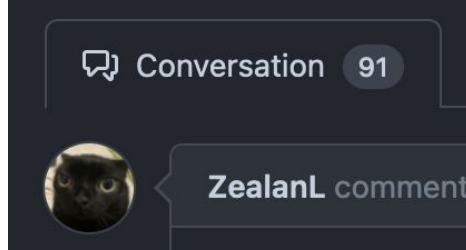
Your program can be used by a bad actor to access and modify arbitrary user memory?



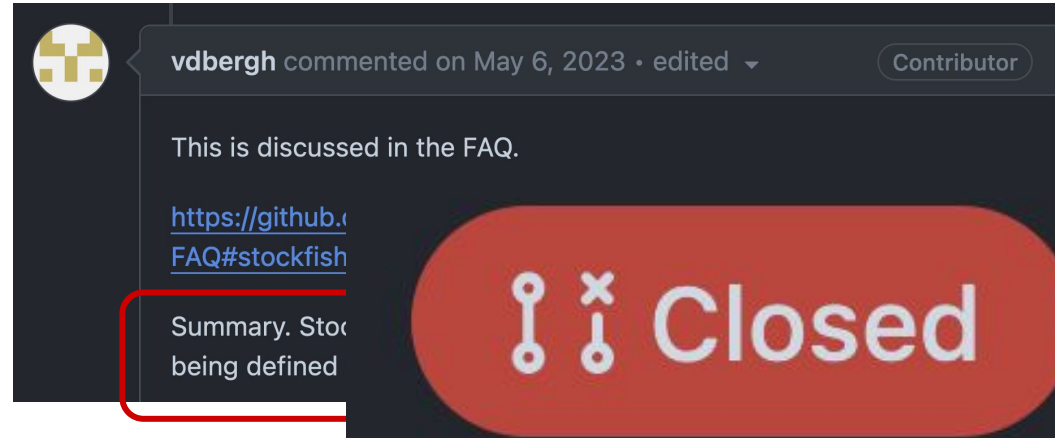
It's a feature, not a bug



# What does this have to do with programming?



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

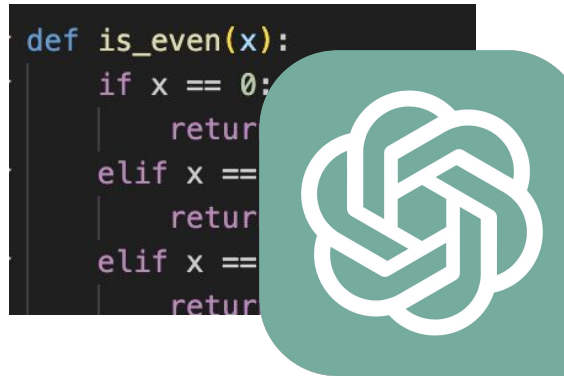


It's a feature, not a bug

# What does this have to do with programming?



Please write a program  
to <do some task>



Ok, here you go



Is this correct?

(Why not?)

# What does this have to do with programming?



Please write a program  
to <do some task>

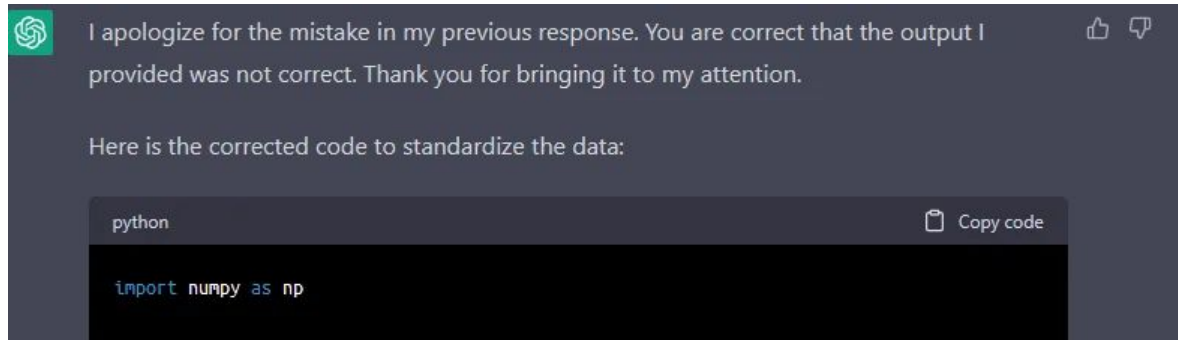


Ok, here you go



Is this correct?

(Why not?)



# What does this have to do with programming?



Please write a program  
to <do some task>



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 ...

# Making the situation worse...

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

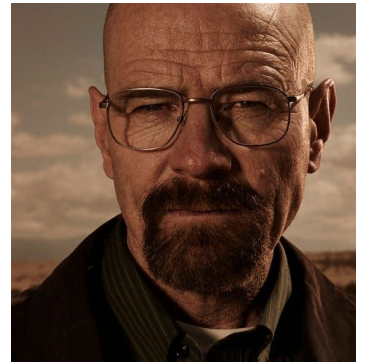


It may require an **esoteric/obscure** input



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.

▲ Ubuntu Bug 255161: Openoffice can't print on Tuesdays (launchpad.net)

244 points by franze on Aug 13, 2014 | [hide](#) | [past](#) | [favorite](#) | [37 comments](#)

Infinite loop heisenbug: it exits if I add a printout

Asked 9 years, 1 month ago   Modified 3 years, 1 month ago   Viewed 2k times

# The problem

We need a

**clear and unambiguous**

way to determine if programs are correct.

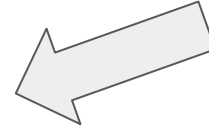
# The problem

We need a

**clear and unambiguous**

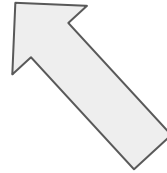
way to determine if programs are correct.

Everyone should agree!



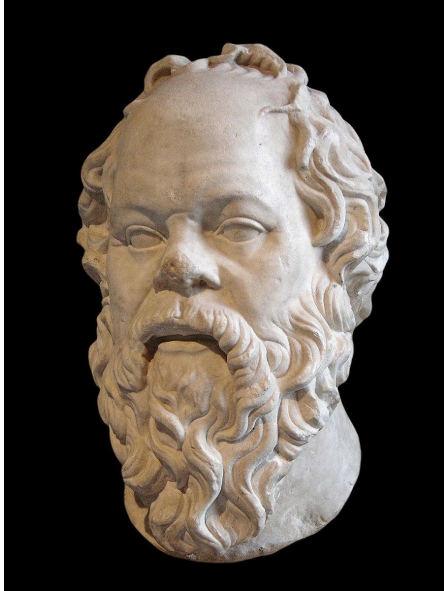
That is:

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





Clear and unambiguous?



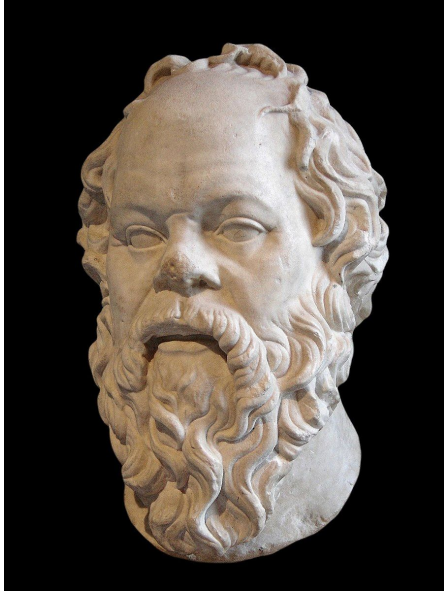
All men are mortal.

Socrates is a man.

Therefore, Socrates is mortal.<sup>[2]</sup>

Logical Syllogism

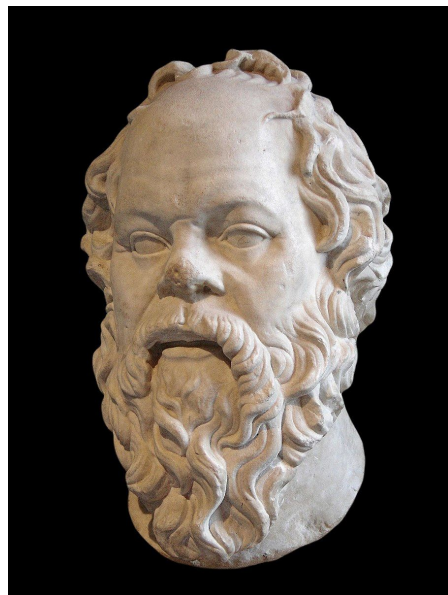
# Clear and unambiguous?



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

even to the most serious skeptic.

# Clear and unambiguous?



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

*Dem.*

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

[\*51·231]  $\equiv . \iota'x \cap \iota'y = \Lambda .$

[\*13·12]  $\equiv . \alpha \cap \beta = \Lambda \quad (1)$

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

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

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

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

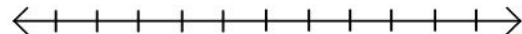
(Don't worry, I won't ask you to write this)

# Everything is Logic



# Proofs can be applied to programs!

Programs can be modeled as mathematical objects [0, 1, 2, 3, ...]

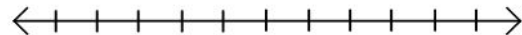


We can test if the program is correct by:

1. Writing down what the program **does (and should do)**
2. Coming up with a **rigorous mathematical** argument

# Proofs can be applied to programs!

Programs can be modeled as mathematical objects  $[0, 1, 2, 3, \dots]$



We can test if the program is correct by:

1. Writing down what the program **does (and should do)**
2. Coming up with a **rigorous mathematical** argument

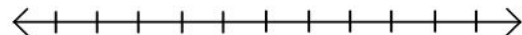


Aside: proofs ARE programs (Curry-Howard 1934, 1969)

[Curry-Howard Correspondence](#)

# Proofs can be applied to programs!

Programs can be modeled as mathematical objects  $[0, 1, 2, 3, \dots]$



We can test if the program is correct by:

1. Writing down what the program **does (and should do)**
- ~~2. Coming up with a **rigorous mathematical** argument~~



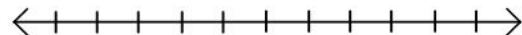
~~Aside: proofs ARE programs (Curry Howard 1934, 1969)~~

~~[Curry Howard Correspondence](#)~~

# Proofs can be applied to programs!

Programs can be modeled as mathematical objects  $[0, 1, 2, 3, \dots]$

We can test if the program is correct by:



1. Writing down what the program does (and should do)
- ~~2. Coming up with a rigorous mathematical argument~~

This class is not about writing proofs

Better: we will make the computer do the math for us!



## ✨ Puzzle ✨

I'm thinking of 4 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?

Slido

Join at  
**slido.com**  
**#2041 532**



# Tools used in industry

## Testing tools



(Many others)

# Tools used in industry

Automated theorem provers



Microsoft  
Research

## Z3

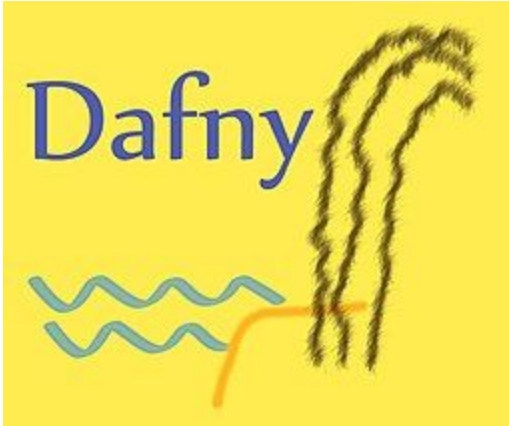
Z3 is a theorem prover from Microsoft Research. It is licensed under the [MIT license](#).



*"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

# Tools used in industry

This class is not about writing proofs

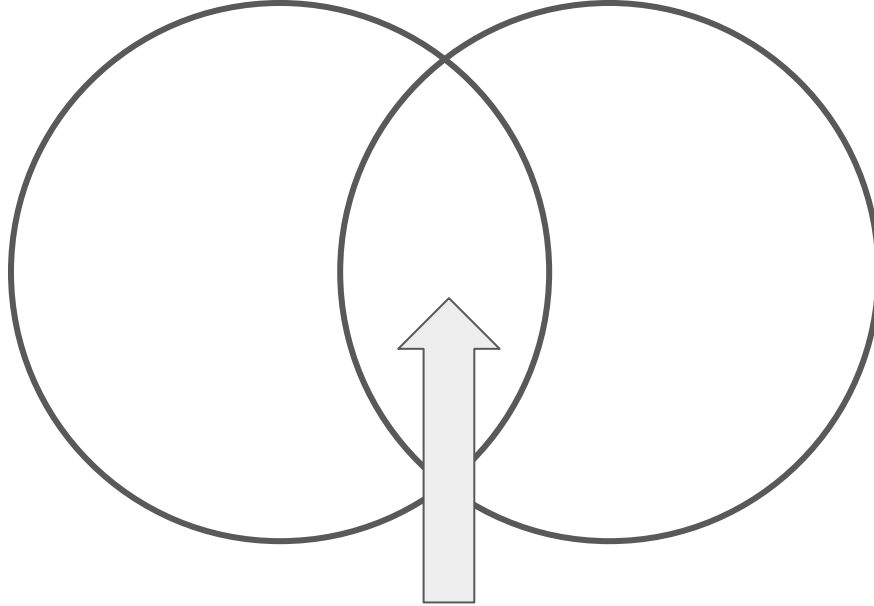
- I am more interested in:
  - showing you the **industrial tools available** that can be used to practically **test and/or demonstrate** software correctness.
  - helping you **think clearly about** what programs you write, why they work, and why they don't



# Areas of computer science

Programming Languages

Formal Methods



This class

# Summary: what this class is about

**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)

Q: What is a special topics course?

A: basically an **experimental** course offering



Q: What is a special topics course?

A: basically an **experimental** course offering



Q: What is a special topics course?

A: basically an **experimental** course offering



Listed title: “Programming languages & Compilers” is a default placeholder

Q: Is this class about programming languages and compilers?

Short answer: **Yes** and **No**

Long answer:

**Yes:** The concepts covered in this course are fundamental to PL and compiler development

**No:** Not a traditional languages or compilers class (No overlap with 140A or 142)

# Q: Are there any prerequisites?

Short answer: No

Long answer:

- I will assume a basic programming background
  - (e.g., ability to write [FizzBuzz](#), ECS 36A/36B/36C)
- Some familiarity with mathematical reasoning (e.g. ECS 20) helpful, but not required

# Q: Is this course right for me?

Short answer: Probably!

Long answer: Especially if:

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

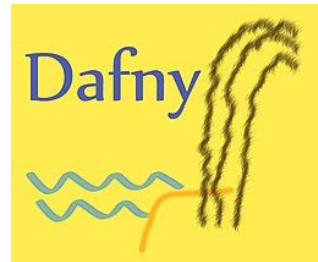


# Plan for today

1. What is this class about?
2. Syllabus and logistics
3. Demo

# Learning objectives

1. Understand the concept of software correctness and its importance.
2. Use random testing tools like Hypothesis for software testing.
3. Use model-based verifiers like Z3 for software analysis.
4. Apply program verification tools such as Dafny to ensure the correctness of software.
5. Explore advanced topics in software correctness, including advanced type systems and concurrency.



# Waitlist

**During the first week:** Please do attend the class

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

**By the end of the second week:** You will be admitted off the waitlist if you have been attending class

- Deadline to drop: 10th day of the quarter/instruction (drop deadline)

# Waitlist

## **During the third week:**

- Enrollment should already be determined by this time
- Official add deadline is ~13th day
- You must be enrolled to continue participating in the course

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

# Grading

- **Homeworks (50%):** 5 assignments (due bi-weekly).
- **Participation (10%):** via in-class polls and Piazza.
- **Final Exam (40%):** covering all topics covered in the course.

# Grading

- **Homeworks (50%):** 5 assignments (due bi-weekly).
- **Participation (10%):** via in-class polls and Piazza.
- **Final Exam (40%):** covering all topics covered in the course.

**Homework 0** (setup + installation) will be due Monday 1 week from today

**Homework 1-5 every ~2 weeks**

# Attendance

There will be 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 will be recorded. There will be a way to make up the in-class polls

# In class polls

I am trying out Slido for now –

this may change depending on how easy it is to export the names

Join at  
**slido.com**  
**#2041 532**





# Piazza

189C

# Discord

Will be posted later today on Piazza

# AI Policy

- Allowed and encouraged! (But not required)
- [Advice from Jason Lowe-Power](#)
- Final exam will be in-class and closed-book

# Collaboration Policy

- Allowed and encouraged!
- Please list your collaborators at the top of your homework
- Everyone should submit their own solution

# Schedule

[Schedule](#)

# Communication

TA: **Parnian Kamran**

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

Questions for me



# Demos

[Lecture 0 Demos](#)