

# CS 472 – Provably Correct Programming

William Mansky

## Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

# Welcome to the Course!

- This is CS 472, Provably Correct Programming
- I'm glad you're here!
- Meets TR 11:00 AM – 12:15 PM in TBH 180G
- My office hours: Tuesday 2 PM, Friday 11 AM, and by appointment, in SEO 1331 and on Zoom via Blackboard
- TA office hours: TBA
  - Office hours are great for homework help, or just to say hi!

# Course Information

- Professor: William Mansky (he/him) ([mansky1@uic.edu](mailto:mansky1@uic.edu))
- TA: Eli Whitehouse ([ewhit37@uic.edu](mailto:ewhit37@uic.edu))
- Prerequisites: CS 301 (logic and proofs), CS 341 (functional programming)
- Website: <https://www.cs.uic.edu/~mansky/teaching/cs472/sp25/>
- Anonymous in-class questions: <https://pollev.com/wmansky771>
- In-person lectures, recordings (probably incomplete) on [Blackboard](#)
- Discussion board on [Piazza](#), assignments via [Gradescope](#) (entry code PY2EGN)

# Asking questions

- In class: raise your hand anytime
- You can ask questions anonymously with PollEverywhere (<https://pollev.com/wmansky771>)
- On [Piazza](#)
  - Can ask/answer anonymously
  - Can post privately to instructors
  - Can answer other students' questions
- In office hours
- If you have a question, someone else probably has the same question!

## Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

# Software works badly



**United Airlines says the outage that held up departing flights was not a cybersecurity issue**

NASA engineers finally came to the conclusion that there were too many files on the file system.  
[https://en.wikipedia.org/wiki/Spirit\\_\(rover\)](https://en.wikipedia.org/wiki/Spirit_(rover))

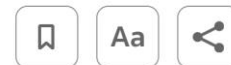


...to grow large enough to overwrite data that could cause unintended acceleration  
[https://en.wikipedia.org/wiki/2009%E2%80%932011\\_Toyota\\_vehicle\\_recalls](https://en.wikipedia.org/wiki/2009%E2%80%932011_Toyota_vehicle_recalls)

## Explainer: What happened to shut down Toyota's production in Japan?

Reuters

August 30, 2023 6:27 AM CDT · Updated 4 months ago



## Knight Capital Says Trading Glitch Cost It \$440 Million

BY NATHANIEL POPPER AUGUST 2, 2012 9:07 AM 356

Runaway Trades Spread Turmoil Across Wall St.



[https://www.nytimes.com/2012/08/02/business/knight-capital-says-trading-glitch-cost-it-440-million.html](#)



[https://www.nytimes.com/2012/08/02/business/knight-capital-says-trading-glitch-cost-it-440-million.html](#)

# Can we change the way we program?

## STOP DOING JAVASCRIPT

- DOCUMENTS WERE NOT SUPPOSED TO BE TURING-COMPLETE
- YEARS OF WEB TECHNOLOGY yet NO REAL-WORLD USE FOUND for doing more than HTML FORMS
- Wanted to do more anyway for a laugh? We had a tool for that: It was called “ADOBE FLASH”
- “Yes please connect the Redux thunk to a Suspense. Please instantiate WebAssembly streaming.” - Statements dreamed up by the utterly Deranged

LOOK at what Web Developers have been demanding your Respect for all this time with all the computers & compilers we built for them

(This is REAL Javascript, done by REAL Web Developers)

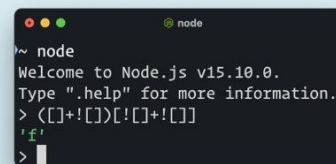


?????



node\_modules  
807 items

???????



????????????????????

“Hello I would like `undefined` apples please”

They have played us for absolute fools

# Can we change the way we program?

- Programs in safer languages (Rust, OCaml, ...) still have bugs
- Most programmers don't get to choose what language they write in!
  - Need to maintain/interoperate with existing code
  - We mostly write in the languages we know

# Can we change the way we program?

- Prove programs correct!

```
i = 1;
while(i <= n) {
    r = r * i;
    i++;
}
```



This course:

How can we write these proofs?

How can we write programs so they're easier to prove?

- This program will:
  - never overflow its stack
  - never dereference a null pointer
  - never call a function without meeting its preconditions
  - always return the right result!

## Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

# Grading

- In-class exercises: 25%
- Assignments: 50%
- Project: 25%

# Exercises

- In each class, we'll work through some example problems/proofs
- Submit via [Gradescope](#)
- Due at the start of the next class
- You get credit as long as you make some progress on the problem
- Feel free to discuss with your neighbors, ask questions, suggest other approaches, etc.!

# Assignments

- Programming/proving assignments
- Submit via [Gradescope](#)
- Due at 11:59 PM on the due date
- You can discuss strategy with other students, but don't look at each other's code!
- Cite your sources (websites, other students, StackOverflow, ChatGPT, etc.)
- You'll get most of the credit for attempting a problem, even if you don't finish it – do what you can, and we'll work through tricky ones in class after the deadline

# Online Sources and ChatGPT

- You *can* find solutions online, but:
  1. It'll make the class harder; the assignments are there to help you learn.
  2. Always cite your sources! (websites, GPT, etc.)
  3. Please don't make your solutions public (e.g. on GitHub)
  4. There's less theorem-prover code out there than other kinds of code, so what you find will often be outdated or unreliable
    - a. ChatGPT is a *plagiarism machine*, so google your code!
  5. Always *run your proofs* and see if they work! (and if they don't, don't submit them!)

## Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

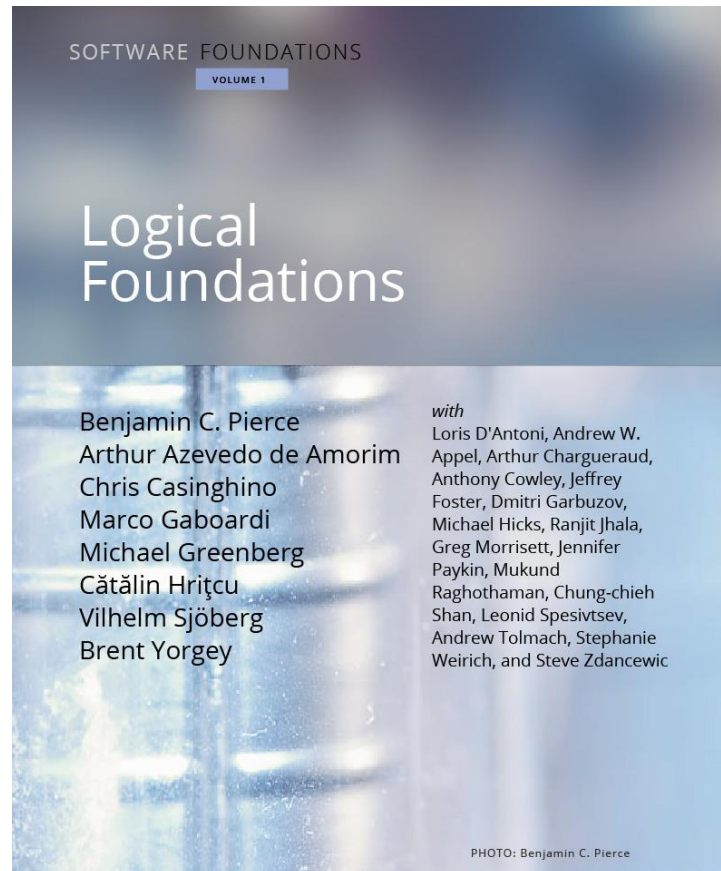
# Getting Started with Proofs

- You already know how to write programs: languages, compilers, IDEs, etc.
- How do we write proofs? The same way: with software tools!
- Tool #1: The Coq/Rocq proof assistant (<https://coq.inria.fr/>)
- For the first ~4 weeks, we'll learn how to use it to write guaranteed-correct mathematical proofs
- After that, we'll apply those techniques to programs!
- And look at some complicated use cases, esp. *concurrent* programs (see also CS 454)

# Interactive Theorem Provers

- In the theorem prover, we can:
  1. Write **definitions**, in a math-like programming language
  2. Write **proofs** about those definitions, using logic “tactics”
  3. See the **proof state** at each point in a proof (what do we know? what do we still need to show?)
  4. Automatically **check** that each step of our proofs is valid

# Logical Foundations



- Online textbook, each chapter is a file that can be run in Coq/Rocq
- Contents:
  - Introduction to theorem proving
  - Basic logic and functional programming
  - More advanced logic, mostly induction
  - How to describe the behavior of programs

<https://softwarefoundations.cis.upenn.edu/lf-current/index.html>

# Getting Started with Coq/Rocq

- Available online at <https://coq.inria.fr/>
- You can download installers for Windows and Mac from the website
- File extension is `.v`
- If it matters, we'll use version 8.19.2
- Two main IDEs: CoqIDE (ships with Coq) and Visual Studio Code (VSCoq extension)

# Today's Exercise

1. Download and install Coq (<https://github.com/coq/platform/releases/tag/2024.10.1>, or from the download links at <https://coq.inria.fr/> or [rocq-prover.org](http://rocq-prover.org))
2. Download and unpack the textbook (<https://softwarefoundations.cis.upenn.edu/lf-current/index.html>)
3. Run **make** in the textbook's folder to compile the textbook. If you don't have a command line with **make**, you'll need to set one up: I use Cygwin (<https://cygwin.com/install.html>)
  - a) If **make** doesn't work, you might need to add Coq to your PATH
4. If you finish, run [demo.v](#), then submit a screenshot for Exercise 1/14 on Gradescope (the whole file should be green). If you haven't finished, submit a description of where you're stuck instead.

If you get stuck, raise your hand, post on Piazza, or come by office hours

## Questions

Nobody has responded yet.

Hang tight! Responses are coming in.