**Learning objectives:**

- ☐ familiarize yourself with the course objectives: problem-solving!
- ☐ set expectations in class: taking notes, participating, problem sets,
- ☐ set expectations outside class: problem sets, self-grading,
- ☐ identify whether a statement is a proposition,
- ☐ describe some proof terminology: lemma, theorem, corollary.

Welcome to CS 200 :)

I'm glad you're here and hope you're ready for a fun semester filled with puzzles, games and other interesting problems from daily life.

Really! The skills you will develop in this class will be helpful, not only for your future in computer science, but will also help you solve problems in the world around you. I promise you will never look at a chocolate bar, Google Maps, or a deck of cards the same ever again.

## 1 Course objectives

The primary goal of this course is for you to enhance your pen-and-paper problem solving skills for computer science applications. But it isn't enough to just come up with a solution when solving a problem. You need to communicate it to your peers and be (hopefully) confident that your solution is correct. We want you to use mathematical arguments to support your solutions.

**Math? I want to study computer science!**

Yes! Math is the language of science, so one of our goals in the course is to develop the language you need for your future CS courses. Instead of directly focusing on programming to solve problems, we'll focus on the **problem-solving process** itself. The main goal of this course is to make you a *problem solver*.

By the way, I like to put cartoons in the margins of the notes, and I've decided to use Pokémon (which I found royalty-free here). The one above is Bulbasaur (it isn't necessary to know about Pokémon to follow along with the notes). Please keep an eye out for them!

. . . and let me know if you have any favorites (or other cartoon suggestions to use) in our Intro Form.

### 1.1 Common coding techniques

I've seen a variety of techniques used by both experienced and beginner programmers. The top three dangerous ones are listed below.

- The *I don't even know where to start, so I'll just wait until I have an epiphany* method: usually results in a lot of stress and late-night help sessions.
- The *I know the answer should be multiplied by 2, so I'll just multiply by 2 somewhere* method: might work for some program inputs, but will usually fail for general program inputs.
- The *code and hope* method: consists of programming-as-you-go without carefully considering what the goal of the program is.

Now, let's add one more to the list:

- The *plan, organize, implement, check*: this is what we want you to do! CS 200 will give you the tools to plan your solutions and be confident that they are correct.

Here is a more detailed list of objectives that describes what kinds of tools we will develop to help you solve problems:

- Use mathematical notation for basic mathematical objects such as sets and functions correctly and appropriately.
- Write clear, concise, and correct proofs using the following techniques: direct proof, induction, contrapositive, proof by contradiction, proof by counter example, and proof by cases.
- Analyze function growth and algorithm runtime using asymptotic notation and recurrence relations.
- Describe graphs and their properties, prove statements related to graphs, and understand simple graph algorithms.
- Describe and analyze random events using discrete probability.

**How does this fit in with other courses?**

The are other courses in the department that also give you problem-solving skills, such as CS 201 (Data Structures), but that is more focused on implementing the data structures. In CS 200, we want to focus on the math behind our solutions.

## 2 Topics

Here is a list of topics we will cover in this course. Come back to this list throughout the semester and make sure that you are familiar with the concepts as we cover them.

- ☐ Proofs:
  - ☐ Talking math: theorem, lemma, corollary, propositions, predicates, quantifiers, sets.
  - ☐ Techniques: direct, cases, counterexample, contradiction, contrapositive, induction.
- ☐ Linear algebra: vectors, matrices, proofs, operations.
- ☐ Graphs: edge, vertex, adacency matrix, simple, bipartite, coloring, trees, breadth-first and depth-first search, relations.
- ☐ Recurrences: linear, divide-and-conquer, calculating sums.
- ☐ Counting: combinations, permutations.
- ☐ Functions: injective, bijective, surjective, $O$, $o$, $\Theta$, $\omega$, $\Omega$ notation.
- ☐ Probability: event, outcome, sample space, tree method, random variables, expectation, linearity of expectation.

**Proof by induction**

One of the most important techniques you will learn in this course is how to do a proof by induction.

## A contract

The course syllabus (here) provides a lot more details as to what to expect both in and outside of class. Please make sure you review it before proceeding. Let's draw up a contract between you and me (Philip) that we can refer back to throughout the semester.

### My promise as an instructor:

- I will create an inclusive, welcoming learning environment, both in and outside our classroom.
- I will prepare lecture notes that you can use to study the course material.
- My problem sets and exams will be fair but still challenge you to apply your knowledge.
- I will grade things as soon as possible to give you feedback on your work.
- I will hold office hours for asking conceptual questions and clarifications. If you are stuck on a problem, I will try to point you in the right direction, but will not give you the answer.

### Your promise as a student:

- I will make sure that I make time for self-care.
- I will take notes and participate in group discussions in a respectful and kind manner.
- I will come to office hours with specific questions, and be mindful of other students that might want to ask questions as well.
- I will keep up with the material by reading the posted lecture notes and watching the lecture videos (weeks 6-12).
- I will practice applying my knowledge with the in-class problems, and will look for other problems when I think I need more practice on a certain topic.
- I will submit my work on time, and let Philip know at least 48 hours in advance if I need an extension.

**Please take care of yourselves!**

This includes sleeping, eating, exercising, whatever you need to do to be YOU both in and out of the classroom.

Signature: _____ date:
Name: Philip Caplan

Signature: _____ date:
Name:

## 3 What is a proof?

Let's start our discussion about proofs. You may have seen some proofs before, maybe when studying geometry in high school (it's okay if you didn't). What's your definition of a proof?

Bulbasaur (now evolved into Ivysaur) has a nice idea. A proof is a method for ascertaining a *truth*. So we can say that a proof is a way to say that something is either *true* or *false*. This "something" is a *proposition*, defined below.

**A proof is a method for ascertaining a truth.**

This is a great definition. Let's generalize it to a mathematical setting.

**Definition 1.** *A **proposition** is a statement that is either true or false.*

So here's a proposition:

**Proposition 1.** $1 + 1 = 2$

which is true. Here's another one:

**Proposition 2.** *I'm watching you, Wazowski.*

Examples of statements that are *not* propositions might be "What did you do over the break?" or "Pass the salt, please."

Our goal is to make propositions, and decide whether they are true or false, using some mathematical arguments. This leads into our definition of a proof.

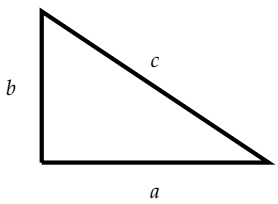
**Definition 2.** *A **mathematical proof** is a verification of a **proposition** by a chain of logical **deductions** from a set of **axioms**.*

We'll look at deductions and axioms soon, but for now just think of axioms as our assumptions that always hold, and deductions as the sequence of steps we use (starting with our axioms, and some other information), to prove something.

But, how do we determine if something is true or false? Some ideas might be:

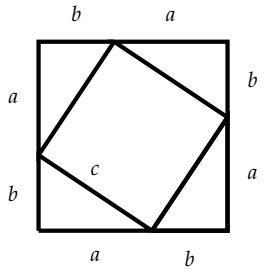
- experimentation & observation,
- counterexamples,
- inner conviction (*I just know it*).

Let's consider a famous example.




**Theorem 1.** *(Pythagorean theorem) Given a planar right triangle with side lengths $a$, $b$ and $c$ – meaning there is one angle of $90°$, which we will take (for generality) to be opposite the side with length $c$. Then $c^2 = a^2 + b^2$.*

Any ideas on how to prove this? Experimenting with different values wouldn't get us very far because we would have to test for an infinite number of values. Coming up with a counterexample is a bit tricky here because a counterexample might be useful to prove something *isn't* true. Well, when coming up with a proof, sometimes you have to think outside the box. In this case, let's think inside the square.




*Proof.* Consider the square with side length $a+b$, formed by placing the same triangle in the arrangement to the right. We know this square has area $(a+b)^2$. Note that this also contains a square with side length $c$ (rotated a little), which has area $c^2$. The area of our triangle is $\frac{1}{2}ab$ and we know that 4 times this area, plus the area of the inner square should equal the area of the outer square:

$$\begin{aligned} 4 \cdot \frac{1}{2}ab + c^2 &= (a+b)^2 \\ 2ab + c^2 &= a^2 + 2ab + b^2 \\ c^2 &= a^2 + b^2 \end{aligned}$$

□

**What is that little box?**

It's common practice to put a little box (□) on the rightmost side of the page to demonstrate that you're done with your proof. Some people write Q.E.D. instead, but we'll use boxes in this course. Make sure you always write a □ after your proof!

To explicitly add a □ in LaTeX math mode (unless you are using a `proof` environment), you can use \Box.

But how did we even know to do that?! Some might argue that because we were looking for something squared, then area (which has units of length-squared) might be useful here. But we assumed a few things about squares and areas. In fact, we were able to derive the Pythagorean theorem, through some axioms of Euclidean geometry.

**Definition 3.** *An **axiom** is a proposition that is assumed to be true.*

In general, we will assume a set of axioms, and derive more complicated things from them. This might be a *theorem* (like we just did with the Pythagorean theorem), a *corollary* or a *lemma*. A *corollary* is a proposition that follows quite easily from a theorem, whereas a *lemma* is a proposition that is useful for proving other propositions. In other words, a lemma might come before a theorem, whereas a corollary comes after it. For example,

**Corollary 1.** *The length of the hypotenuse of a right, isosceles triangle (with the equal side lengths denoted by a) is $c = \sqrt{2}a$.*

*Proof.* By Theorem 1, we know that $c^2 = a^2 + b^2$. For an isosceles triangle, $b = a$, so $c = \sqrt{2}a$. □

Here is another famous theorem.

**Theorem 2.** *There are no positive integers $x, y, z$ such that*

$$x^n + y^n = z^n$$

*for $n > 2$.*

Looks a lot like the Pythagorean theorem, eh? This is known as Fermat's last theorem (conjectured by Pierre de Fermat in 1637), which was finally proved in 1994 by Andrew Wiles. One thing to note is the use of *positive integers*, which leads into our discussion about *sets* (soon).

**Conjectured?**

A **conjecture** is a statement whose truth is still unknown. So technically, it should be called *Fermat's last conjecture* from 1637-1994, but *Fermat's last theorem* after 1994.