Functions
What is a function?
Functions, High-School Edition
\[ f(x) = x^4 - 5x^2 + 4 \]
\[ f(x) = \frac{x^2 + 4x - 9}{x^2 + 10x + 21} \]
Functions, High-School Edition

• In high school, functions are usually given as objects of the form

\[ f(x) = \frac{x^3 + 3x^2 + 15x + 7}{1 - x^{137}} \]

• What does a function do?
  • It takes in as input a real number.
  • It outputs a real number
  • ... except when there are vertical asymptotes or other discontinuities, in which case the function doesn't output anything.
Functions, CS Edition
int flipUntil(int n) {
  int numHeads = 0;
  int numTries = 0;

  while (numHeads < n) {
    if (randomBoolean()) numHeads++;
    numTries++;
  }

  return numTries;
}
Functions, CS Edition

• In programming, functions
  • might take in inputs,
  • might return values,
  • might have side effects,
  • might never return anything,
  • might crash, and
  • might return different values when called multiple times.
What's Common?

- Although high-school math functions and CS functions are pretty different, they have two key aspects in common:
  - They take in inputs.
  - They produce outputs.
- In math, we like to keep things easy, so that's pretty much how we're going to define a function.
Rough Idea of a Function:

A function is an object $f$ that takes in an input and produces exactly one output.

(This is not a complete definition – we'll revisit this in a bit.)
High School versus CS Functions

• In high school, functions usually were given by a rule:
  \[ f(x) = 4x + 15 \]

• In CS, functions are usually given by code:

```c
int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

• What sorts of functions are we going to allow from a mathematical perspective?
... but also ...
\[ f(x) = x^2 + 3x - 15 \]
\[ f(n) = \begin{cases} 
-\frac{n}{2} & \text{if } n \text{ is even} \\
\frac{(n+1)}{2} & \text{otherwise}
\end{cases} \]

Functions like these are called **piecewise functions**.
To define a function, you will typically either

· draw a picture, or
· give a rule for determining the output.
In mathematics, functions are deterministic. That is, given the same input, a function must always produce the same output.

The following is a perfectly valid piece of C++ code, but it’s not a valid function under our definition:

```cpp
int randomNumber(int numOutcomes) {
    return rand() % numOutcomes;
}
```
One Challenge
\[ f(x) = x^2 + 2x + 5 \]

\[ f(3) = 3^2 + 3 \cdot 2 + 5 = 20 \]

\[ f(0) = 0^2 + 0 \cdot 2 + 5 = 5 \]

\[ f(\text{ Pikachu }) = \ldots ? \]
\[ f(\text{apple}) = ? \]

\[ f(137) = \ldots ? \]
We need to make sure we can't apply functions to meaningless inputs.
Domains and Codomains

- Every function $f$ has two sets associated with it: its **domain** and its **codomain**.
- A function $f$ can only be applied to elements of its domain. For any $x$ in the domain, $f(x)$ belongs to the codomain.

![Diagram showing the relationship between domain and codomain]

The function must be defined for every element of the domain.

The output of the function must always be in the codomain, but not all elements of the codomain must be produced as outputs.
Domains and Codomains

- Every function $f$ has two sets associated with it: its **domain** and its **codomain**.
- A function $f$ can only be applied to elements of its domain. For any $x$ in the domain, $f(x)$ belongs to the codomain.

```cpp
double absoluteValueOf(double x) {
    if (x >= 0) {
        return x;
    } else {
        return -x;
    }
}
```

The **domain** of this function is $\mathbb{R}$. Any real number can be provided as input.

The **codomain** of this function is $\mathbb{R}$. Everything produced is a real number, but not all real numbers can be produced.
Domains and Codomains

- If $f$ is a function whose domain is $A$ and whose codomain is $B$, we write $f : A \rightarrow B$.
- Think of this like a “function prototype” in C++.
The Official Rules for Functions

• Formally speaking, we say that $f : A \rightarrow B$ if the following two rules hold.

• First, $f$ must be obey its domain/codomain rules:
  $\forall a \in A. \exists b \in B. f(a) = b$
  ("Every input in $A$ maps to some output in $B$.")

• Second, $f$ must be deterministic:
  $\forall a_1 \in A. \forall a_2 \in A. (a_1 = a_2 \rightarrow f(a_1) = f(a_2))$
  ("Equal inputs produce equal outputs.")

• If you’re ever curious about whether something is a function, look back at these rules and check! For example:
  • Can a function have an empty domain?
  • Can a function have an empty codomain?
Defining Functions

• Typically, we specify a function by describing a rule that maps every element of the domain to some element of the codomain.

• Examples:
  • \( f(n) = n + 1 \), where \( f : \mathbb{Z} \to \mathbb{Z} \)
  • \( g(x) = \sin x \), where \( g : \mathbb{R} \to \mathbb{R} \)

• Notice that we're giving both a rule and the domain/codomain.
Is This a Function From $A$ to $B$?

$A$

Stanford

Berkeley

Michigan

Arkansas

Cardinal

Blue

Gold

White

$B$
Is This a Function From $A$ to $B$?

- California
- New York
- Delaware
- Washington DC

- Dover
- Sacramento
- Albany

$A$ to $B$?
Is This a Function From A to B?
Combining Functions
\( f : People \rightarrow Places \)

- Keith \rightarrow Mountain View
- Josh \rightarrow San Francisco
- Divya \rightarrow Redding, CA
- Teresa \rightarrow Barrow, AK
- Julian \rightarrow Palo Alto

\( g : Places \rightarrow Prices \)

- Mountain View \rightarrow Far Too Much
- San Francisco \rightarrow King's Ransom
- Redding, CA \rightarrow A Modest Amount
- Barrow, AK \rightarrow More Than You’d Expect
- Palo Alto \rightarrow More Than You’d Expect

\( h : People \rightarrow Prices \)

\[ h(x) = g(f(x)) \]
Function Composition

- Suppose that we have two functions $f : A \rightarrow B$ and $g : B \rightarrow C$.
- Notice that the codomain of $f$ is the domain of $g$. This means that we can use outputs from $f$ as inputs to $g$. 

\[ f \circ g \]

\[ f(x) \]

\[ g(f(x)) \]
Function Composition

• Suppose that we have two functions \( f : A \to B \) and \( g : B \to C \).

• The composition of \( f \) and \( g \), denoted \( g \circ f \), is a function where
  
  • \( g \circ f : A \to C \), and
  
  • \( (g \circ f)(x) = g(f(x)) \).

• A few things to notice:
  
  • The domain of \( g \circ f \) is the domain of \( f \). Its codomain is the codomain of \( g \).

  • Even though the composition is written \( g \circ f \), when evaluating \( (g \circ f)(x) \), the function \( f \) is evaluated first.

The name of the function is \( g \circ f \). When we apply it to an input \( x \), we write \( (g \circ f)(x) \). I don't know why, but that's what we do.
Time-Out for Announcements!
Applications are now open for Google’s 2019 student scholarships. Selected student will each receive a $10,000 scholarship for the 2019-2020 academic year and be invited to attend the annual Google Scholars' Retreat at the Googleplex next summer.

Here is a list of open scholarships that you can share with your students:

**Women Techmakers Scholars Program** ([https://www.womentechmakers.com/scholars](https://www.womentechmakers.com/scholars)) is open to current undergraduate or graduate students who will be studying at a university in the United States or Canada for the 2019-2020 academic year. We strongly encourage people who identify as female to apply. Deadline: December 6, 2018

**Generation Google Scholarship** ([https://buildyourfuture.withgoogle.com/scholarships/generation-google-scholarship/#!?detail-content-tabby_activeEl=detail-overview-content](https://buildyourfuture.withgoogle.com/scholarships/generation-google-scholarship/#!?detail-content-tabby_activeEl=detail-overview-content)) is open to current undergraduate or graduate students who will be studying at a university in the United States or Canada for the 2019-2020 academic year. We strongly encourage students from historically underrepresented groups, including Black/African American, Hispanic/Latino, American Indian, or Filipino/Native Hawaiian/Pacific Islander, to apply. Deadline: December 6, 2018

**Google Lime Scholarship** ([https://buildyourfuture.withgoogle.com/scholarships/google-lime-scholarship/#!?detail-content-tabby_activeEl=detail-overview-content](https://buildyourfuture.withgoogle.com/scholarships/google-lime-scholarship/#!?detail-content-tabby_activeEl=detail-overview-content)) is open to current undergraduate or graduate students with disabilities who will be studying at a university in the United States or Canada for the 2019-2020 academic year. Deadline: December 9, 2018

**Google Student Veterans of America Scholarship** ([https://buildyourfuture.withgoogle.com/scholarships/google-student-veterans-of-america-scholarship/#!?detail-content-tabby_activeEl=detail-overview-content](https://buildyourfuture.withgoogle.com/scholarships/google-student-veterans-of-america-scholarship/#!?detail-content-tabby_activeEl=detail-overview-content)) is open to current undergraduate or graduate student veterans who will be studying at a university in the United States for the 2019-2020 academic year. Deadline: November 1, 2018

For questions and complete details on all of our scholarships, please visit [https://buildyourfuture.withgoogle.com/scholarships/s](https://buildyourfuture.withgoogle.com/scholarships/s).
Problem Set Three

• Problem Set Two was due today at 2:30PM.
  • Want to use late days? Submit the assignment by Sunday at 2:30PM.

• Problem Set Three goes out today.
  • The checkpoint is due on Monday at 2:30PM.
  • The remaining problems are due Friday at 2:30PM.
  • Play around with binary relations, functions, their properties, and their applications!

• As usual, feel free to ask questions!
  • Ask on Piazza!
  • Stop by office hours!
Extra Practice Problems

• We’ve posted a set of ten extra practice problems to the course website, with solutions.
• Feel free to use these to help solidify your understanding of concepts or to practice techniques.
• We’re happy to discuss these questions in office hours – there’s no “stakes” on them.
Your Questions
“I went to the career fair and just felt so inadequate after. How do you keep from letting those situations get to you?”

“How should I decide between working for a smaller startup or a big company? How much do resumes matter in the tech industry?”

“Are career fairs helpful and necessary? I didn’t go because I was scared. I worry if I never attend one it will be difficult to get a job. What’s do you think?”

I’m going to give a meta-answer to this one!
“Can you answer a more diverse set of questions on here beyond career advice? This area seems to be overrepresented, and there's many interesting others!”

Yes, absolutely! I’ll turn that back around and ask for all y’all to ask questions on other topics as well.
“How do I feel like I can belong in CS? I find it really hard when all of the CS professors I've had at Stanford are men”

I’m going to split this into two separate questions – one about belonging and one about faculty diversity – rather than take on the two together. Want to talk about the two in tandem? Come talk to me in office hours, or email me and let’s set up a time to chat one-on-one. I want to hear your thoughts!

We have some stellar female faculty that we’ve hired in the past few years and I’d be happy to make introductions if you’d like. You can also get involved in the hiring process; the department solicits input from students about faculty candidates.

As for community – the fact that CS is so big cuts both ways. The major is fairly heterogeneous and there are lots of communities to reach out to. Fun fact – there are more women majoring in CS than in any other major at Stanford. Check out groups like WiCS, SOLE, SBSE, etc. if this is something you’re interested in!
Back to CS103!
Special Types of Functions
Injective Functions

- A function \( f : A \rightarrow B \) is called **injective** (or **one-to-one**) if the following statement is true about \( f \):

  \[
  \forall a_1 \in A. \forall a_2 \in A. (a_1 \neq a_2 \rightarrow f(a_1) \neq f(a_2))
  \]

  (“If the inputs are different, the outputs are different.”)

- The following first-order definition is equivalent and is often useful in proofs.

  \[
  \forall a_1 \in A. \forall a_2 \in A. (f(a_1) = f(a_2) \rightarrow a_1 = a_2)
  \]

  (“If the outputs are the same, the inputs are the same.”)

- A function with this property is called an **injection**.

- How does this compare to our second rule for functions?
Injective Functions

**Theorem:** Let $f : \mathbb{N} \to \mathbb{N}$ be defined as $f(n) = 2n + 7$. Then $f$ is injective.

**Proof:**

What does it mean for the function $f$ to be injective?

\[
\forall n_1 \in \mathbb{N}. \forall n_2 \in \mathbb{N}. \ (f(n_1) = f(n_2) \rightarrow n_1 = n_2)
\]
\[
\forall n_1 \in \mathbb{N}. \forall n_2 \in \mathbb{N}. \ (n_1 \neq n_2 \rightarrow f(n_1) \neq f(n_2))
\]

Therefore, we'll pick arbitrary $n_1, n_2 \in \mathbb{N}$ where $f(n_1) = f(n_2)$, then prove that $n_1 = n_2$. 

Injective Functions

**Theorem:** Let \( f : \mathbb{N} \to \mathbb{N} \) be defined as \( f(n) = 2n + 7 \). Then \( f \) is injective.

**Proof:** Consider any \( n_1, n_2 \in \mathbb{N} \) where \( f(n_1) = f(n_2) \). We will prove that \( n_1 = n_2 \).

Since \( f(n_1) = f(n_2) \), we see that

\[
2n_1 + 7 = 2n_2 + 7.
\]

This in turn means that

\[
2n_1 = 2n_2
\]

so \( n_1 = n_2 \), as required. ■

**Good exercise:** Repeat this proof using the other definition of injectivity!
Injective Functions

**Theorem:** Let \( f : \mathbb{Z} \to \mathbb{N} \) be defined as \( f(x) = x^4 \). Then \( f \) is not injective.

**Proof:**

What does it mean for \( f \) to be injective?

\[
\forall x_1 \in \mathbb{Z}. \forall x_2 \in \mathbb{Z}. \ (x_1 \neq x_2 \rightarrow f(x_1) \neq f(x_2))
\]

What is the negation of this statement?

\[
\neg \forall x_1 \in \mathbb{Z}. \forall x_2 \in \mathbb{Z}. \ (x_1 \neq x_2 \rightarrow f(x_1) \neq f(x_2))
\]

\[
\exists x_1 \in \mathbb{Z}. \exists x_2 \in \mathbb{Z}. \ (x_1 \neq x_2 \land \neg(f(x_1) \neq f(x_2)))
\]

Therefore, we need to find \( x_1, x_2 \in \mathbb{Z} \) such that \( x_1 \neq x_2 \), but \( f(x_1) = f(x_2) \). Can we do that?
Injective Functions

**Theorem:** Let $f: \mathbb{Z} \rightarrow \mathbb{N}$ be defined as $f(x) = x^4$. Then $f$ is not injective.

**Proof:** We will prove that there exist integers $x_1$ and $x_2$ such that $x_1 \neq x_2$, but $f(x_1) = f(x_2)$.

Let $x_1 = -1$ and $x_2 = +1$.

\[
f(x_1) = f(-1) = (-1)^4 = 1
\]

and

\[
f(x_2) = f(1) = 1^4 = 1,
\]

so $f(x_1) = f(x_2)$ even though $x_1 \neq x_2$, as required. ■
Injections and Composition
Injections and Composition

- **Theorem:** If $f : A \rightarrow B$ is an injection and $g : B \rightarrow C$ is an injection, then the function $g \circ f : A \rightarrow C$ is an injection.

- Our goal will be to prove this result. To do so, we're going to have to call back to the formal definitions of injectivity and function composition.
**Theorem:** If $f : A \to B$ is an injection and $g : B \to C$ is an injection, then the function $g \circ f : A \to C$ is also an injection.

**Proof:** Let $f : A \to B$ and $g : B \to C$ be arbitrary injections. We will prove that the function $g \circ f : A \to C$ is also injective.

There are two definitions of injectivity that we can use here:

\[
\forall a_1 \in A. \forall a_2 \in A. ((g \circ f)(a_1) = (g \circ f)(a_2) \rightarrow a_1 = a_2)
\]

\[
\forall a_1 \in A. \forall a_2 \in A. (a_1 \neq a_2 \rightarrow (g \circ f)(a_1) \neq (g \circ f)(a_2))
\]

Therefore, we’ll choose an arbitrary $a_1, a_2 \in A$ where $a_1 \neq a_2$, then prove that $(g \circ f)(a_1) \neq (g \circ f)(a_2)$.
**Theorem:** If \( f : A \to B \) is an injection and \( g : B \to C \) is an injection, then the function \( g \circ f : A \to C \) is also an injection.

**Proof:** Let \( f : A \to B \) and \( g : B \to C \) be arbitrary injections. We will prove that the function \( g \circ f : A \to C \) is also injective. To do so, consider any \( a_1, a_2 \in A \) where \( a_1 \neq a_2 \). We will prove that \((g \circ f)(a_1) \neq (g \circ f)(a_2)\).

How is \((g \circ f)(x)\) defined?

\[
(g \circ f)(x) = g(f(x))
\]

So we need to prove that \( g(f(a_1)) \neq g(f(a_2)) \).
**Theorem:** If $f : A \to B$ is an injection and $g : B \to C$ is an injection, then the function $g \circ f : A \to C$ is also an injection.

**Proof:** Let $f : A \to B$ and $g : B \to C$ be arbitrary injections. We will prove that the function $g \circ f : A \to C$ is also injective. To do so, consider any $a_1, a_2 \in A$ where $a_1 \neq a_2$. We will prove that $(g \circ f)(a_1) \neq (g \circ f)(a_2)$. Equivalently, we need to show that $g(f(a_1)) \neq g(f(a_2))$.

Since $f$ is injective and $a_1 \neq a_2$, we see that $f(a_1) \neq f(a_2)$. Then, since $g$ is injective and $f(a_1) \neq f(a_2)$, we see that $g(f(a_1)) \neq g(f(a_2))$, as required. ■
Another Class of Functions
Surjective Functions

- A function $f : A \rightarrow B$ is called **surjective** (or **onto**) if this first-order logic statement is true about $f$:

  $\forall b \in B. \exists a \in A. f(a) = b$

  ("For every output, there's an input that produces it")

- A function with this property is called a **surjection**.

- How does this compare to our first rule of functions?
Surjective Functions

**Theorem:** Let $f : \mathbb{R} \to \mathbb{R}$ be defined as $f(x) = x / 2$. Then $f(x)$ is surjective.

**Proof:**

What does it mean for $f$ to be surjective?

\[ \forall y \in \mathbb{R}. \; \exists x \in \mathbb{R}. \; f(x) = y \]

Therefore, we'll choose an arbitrary $y \in \mathbb{R}$, then prove that there is some $x \in \mathbb{R}$ where $f(x) = y$. 
Surjective Functions

**Theorem:** Let $f : \mathbb{R} \to \mathbb{R}$ be defined as $f(x) = x / 2$. Then $f(x)$ is surjective.

**Proof:** Consider any $y \in \mathbb{R}$. We will prove that there is a choice of $x \in \mathbb{R}$ such that $f(x) = y$.

Let $x = 2y$. Then we see that

$$f(x) = f(2y) = 2y / 2 = y.$$ 

So $f(x) = y$, as required. ■
Composing Surjections
**Theorem:** If \( f : A \to B \) is surjective and \( g : B \to C \) is surjective, then \( g \circ f : A \to C \) is also surjective.

**Proof:** Let \( f : A \to B \) and \( g : B \to C \) be arbitrary surjections. We will prove that the function \( g \circ f : A \to C \) is also surjective.

What does it mean for \( g \circ f : A \to C \) to be surjective?

\[
\forall c \in C. \exists a \in A. (g \circ f)(a) = c
\]

Therefore, we’ll choose arbitrary \( c \in C \) and prove that there is some \( a \in A \) such that \( (g \circ f)(a) = c \).
**Theorem:** If \( f : A \to B \) is surjective and \( g : B \to C \) is surjective, then \( g \circ f : A \to C \) is also surjective.

**Proof:** Let \( f : A \to B \) and \( g : B \to C \) be arbitrary surjections. We will prove that the function \( g \circ f : A \to C \) is also surjective. To do so, we will prove that for any \( c \in C \), there is some \( a \in A \) such that \( (g \circ f)(a) = c \). Equivalently, we will prove that for any \( c \in C \), there is some \( a \in A \) such that \( g(f(a)) = c \).

Consider any \( c \in C \). Since \( g : B \to C \) is surjective, there is some \( b \in B \) such that \( g(b) = c \). Similarly, since \( f : A \to B \) is surjective, there is some \( a \in A \) such that \( f(a) = b \). Then we see that

\[
g(f(a)) = g(b) = c,
\]

which is what we needed to show. ■
Injections and Surjections

• An injective function associates \textit{at most} one element of the domain with each element of the codomain.

• A surjective function associates \textit{at least} one element of the domain with each element of the codomain.

• What about functions that associate \textit{exactly one} element of the domain with each element of the codomain?
Bijections

- A function that associates each element of the codomain with a unique element of the domain is called **bijective**.
  - Such a function is a **bijection**.
- Formally, a bijection is a function that is both **injective** and **surjective**.
- Bijections are sometimes called **one-to-one correspondences**.
  - Not to be confused with “one-to-one functions.”
Bijections and Composition

• Suppose that \( f : A \rightarrow B \) and \( g : B \rightarrow C \) are bijections.

• Is \( g \circ f \) necessarily a bijection?

• **Yes!**
  
  • Since both \( f \) and \( g \) are injective, we know that \( g \circ f \) is injective.
  
  • Since both \( f \) and \( g \) are surjective, we know that \( g \circ f \) is surjective.
  
  • Therefore, \( g \circ f \) is a bijection.
Inverse Functions
Inverse Functions

• In some cases, it's possible to “turn a function around.”

• Let \( f : A \rightarrow B \) be a function. A function \( f^{-1} : B \rightarrow A \) is called an **inverse of \( f \)** if the following first-order logic statements are true about \( f \) and \( f^{-1} \)

\[
\forall a \in A. \ (f^{-1}(f(a)) = a) \quad \forall b \in B. \ (f(f^{-1}(b)) = b)
\]

• In other words, the functions \( f \) and \( f^{-1} \) undo one another.

• Not all functions have inverses (we just saw a few examples of functions with no inverses).

• If \( f \) is a function that has an inverse, then we say that \( f \) is **invertible**.
Inverse Functions

- **Theorem:** Let $f : A \rightarrow B$. Then $f$ is invertible if and only if $f$ is a bijection.

- These proofs are in the course reader. Feel free to check them out if you'd like!

- **Really cool observation:** Look at the formal definition of a function. Look at the rules for injectivity and surjectivity. Do you see why this result makes sense?
Where We Are

• We now know
  • what an injection, surjection, and bijection are;
  • that the composition of two injections, surjections, or bijections is also an injection, surjection, or bijection, respectively; and
  • that bijections are invertible and invertible functions are bijections.

• You might wonder why this all matters. Well, there's a good reason...
Next Time

• **Cardinality, Formally**
  • How do we rigorously define the idea that two sets have the same size?

• **The Nature of Infinity**
  • It’s even weirder than you think!

• **Cantor’s Theorem Revisited**
  • A formal proof of a major result!