# Problem Set 1

Here we are – the first problem set of the quarter! This problem set is designed to give you practice writing proofs on a variety of different topics. We hope that this problem set gives you a sense of what proof-based mathematics is like and helps solidify your understanding of set theory.

Before you start this problem set, please do the following:

- Review the office hours timetable to find good times to drop on in to ask questions.
- Review Handout #04, "Guide to Set Theory Proofs," for more information about how to structure proofs about sets and their properties.
- Review Handout #05, "Guide to Indirect Proofs," which provides some guidance about how to set up proofs by contradiction and contrapositive.
- Review Handout #06, "Ten Techniques to Get Unstuck," for advice about how to make progress on these sorts of problems when you're not sure what to do.
- Review Handout #07, "Proofwriting Checklist," for a detailed set of criteria you should apply to your proofs before submitting them. ***We will be running this same checklist on your proofs when grading, so please be sure to look over it before submitting!***
- Review the online "Guide to $\in$ and $\subseteq$" to make sure you understand the distinction between these terms.

As always, please feel free to call into office hours or post on EdStem if you have any questions. We're happy to help out.

Good luck, and have fun!

**Due Friday, September 25th at 12:00PM (noon) Pacific time.**

***You are required to abide by the Stanford Honor Code. Information about how the Honor Code applies in CS103 can be found in the "CS103 and the Stanford Honor Code" handout available on the course website.***

## Problem One: Much Ado About Nothing

It can take a bit of practice to get used to the empty set. This problem will ask you to think about a few different sets related to Ø.

Go to the CS103 website and download the starter files for Problem Set One. Unpack the files somewhere convenient and open up the bundled project. Answer each part of this question by editing the file `MuchAdoAboutNothing.sets` in the `res/` directory. There's information in the top of each of the files about how to represent sets; most importantly, note that to write out the empty set, you should write `{}` rather than using the empty-set symbol. For example, the set {Ø} would be written as `{{}}`.

i. Give a set equal to Ø ∪ {Ø}.
ii. Give a set equal to Ø ∩ {Ø}.
iii. Give a set equal to {Ø} ∪ {{Ø}}.
iv. Give a set equal to {Ø} ∩ {{Ø}}.
v. Give a set equal to ℘(℘(Ø)).
vi. Give a set equal to ℘(℘(℘(Ø))).

The starter code contains a driver program you can use to see the contents of your files and run tests to check your answers. Submit your answers through GradeScope under "Coding Problems for Problem Set One" by uploading your edited files. You're welcome to submit as many times as you'd like; we'll count as your grade the last version of the uploaded files.

## Problem Two: Set Theory in C++

The C++ standard library contains a type called `std::set` that represents a set of elements, all of which must be of the same type. For example, the type `std::set<int>` represents a set of integers, the type `std::set<std::string>` represents a set of strings, and `std::set<std::set<int>>` is a type representing a set of sets of integers.

There are all sorts of operations you can perform on `std::set`s. For example, here's how you iterate over all the elements of a set:

```
std::set<T> mySet;
for (T elem: mySet) {
    /* … do something with the current element elem … */
}
```

Here's how you check whether a particular value is an element of a set:

```
if (mySet.count(value)) {
    /* … value ∈ mySet … */
} else {
    /* … value ∉ mySet … */
}
```

And, finally, here's how you can get the cardinality of a set:

```
size_t size = mySet.size();
```

Here, the `size_t` type is a type representing a natural number, since sets can't have negative size. (The folks who designed the C++ standard libraries had a strong discrete math background.)

One of the major differences between the sets we've talked about in CS103 and the `std::set` type is that in discrete mathematics, sets can contain anything – numbers, philosophies, recipes, other sets, etc. – but in C++ all objects in a set must have the same type. For the purposes of this problem, we've created a custom C++ type called `Object`. Variables of type `Object` can represent just about anything, so a `std::set<Object>` represents something pretty similar to the sets we've been studying so far.

Some `Object`s are actually just `std::set`s in disguise. If you have an `Object`, you can test whether it's actually a set by using this provided helper function:

```
bool isSet(Object o);
```

This takes in an `Object`, then returns true if that `Object` is actually a set and false otherwise. If you have an `Object` that really is a set, you can convert it to a set by using this helper function:

```
std::set<Object> asSet(Object o);
```

This function takes in an `Object` that you know happens to be a set, then returns the `std::set<Object>` that it actually is.

For example, suppose you have an `Object` that you know is really the set {1, 2, 3, 4}. You could iterate over it using this code:

```
Object reallyASet = /* … */;
for (Object x: asSet(reallyASet)) {
    /* … do something with x … */
}
```

In this problem, we'd like you to demonstrate your understanding of sets and set theory by coding up a number of functions in C++ that operate on sets. In doing so, we hope that you'll solidify your grasp of the distinctions between related concepts in set theory, such as the the ∈ and ⊆ relations and power sets.

*(Continued on the next page)*

Open the file `SetTheory.cpp` from the starter files. There, you'll find a bunch of stubs of functions that you'll need to implement. The provided starter code contains a test harness you can use to try out your functions. You won't need to modify any of the other C++ files bundled with the starter code.

As with Problem One, you'll submit the code that you write through GradeScope separately from the rest of the problems on this problem set. The GradeScope autograder will get back to you with feedback about how you're doing on this problem, and you're welcome to submit as many times as you'd like.

i. Implement a function

```
bool isElementOf(Object S, Object T);
```

that takes as input two `Object`s `S` and `T`, then returns whether $S \in T$.

*`S` and `T` might not be sets; you'll need to use the `isSet` and `asSet` functions appropriately.*

ii. Implement a function

```
bool isSubsetOf(Object S, Object T);
```

that takes as input an object `S` and an object `T`, then returns whether $S \subseteq T$.

*`S` and `T` might not be sets; use the `isSet` predicate to check whether the appropriate arguments are sets and `asSet` to get a view of them as sets.*

iii. Implement a function

```
bool areDisjointSets(Object S, Object T);
```

that takes as input two objects `S` and `T`, then returns whether `S` and `T` are sets where $S \cap T = \varnothing$. (Two sets with this property are called ***disjoint***.) The input parameters `S` and `T` may or may not be sets, and if they aren't, your function should return false.

iv. Implement a function

```
bool isSingletonOf(Object S, Object T);
```

that takes as input two objects `S` and `T`, then returns whether $S = \{T\}$. Again, `S` and `T` may or may not be sets.

v. Implement a function

```
bool isElementOfPowerSet(Object S, Object T);
```

that takes as input two objects `S` and `T`, then returns whether `S` and `T` are sets and $S \in \wp(T)$. Again, `S` and `T` may or may not be sets.

*As a hint, you shouldn't need to write code that computes $\wp(T)$ explicitly. See if you can find a different way to do this.*

vi. Implement a function

```
bool isSubsetOfPowerSet(Object S, Object T);
```

that takes as input two objects `S` and `T`, then returns whether `S` and `T` are sets and $S \subseteq \wp(T)$. Again, `S` and `T` may or may not be sets.

vii. Implement a function

```
bool isSubsetOfDoublePowerSet(Object S, Object T);
```

that takes as input two objects `S` and `T`, then returns whether `S` and `T` are sets and $S \subseteq \wp(\wp(T))$. Again, `S` and `T` may or may not be sets.

To submit your work, upload your edited `SetTheory.cpp` file to GradeScope. You'll get immediate feedback on your score from our autograder, and you're allowed to resubmit as many times as you'd like. As a note, you shouldn't edit `SetTheory.h`; our autograder will always use the default version of that file when running tests.

## Problem Three: Describing the World in Set Theory

The notation of set theory (e.g. $\cup$, $\cap$, $\wp$, $\subseteq$, $\in$, $\varnothing$, $=$, etc.) is a great tool for describing the real world. Answer each of the following questions by writing an expression using set theory notation, but ***without*** using plain English, ***without*** using set-builder notation, ***without*** introducing any new variables, and ***without*** using propositional or first-order logic (which we'll cover next week). No justification is required.

i. In the Talking Heads song *Crosseyed and Painless*, David Byrne speaks the following lines:

> "Facts are simple and facts are straight.
> Facts are lazy and facts are late."

Let $F$ be the set of all facts. Let $A$, $B$, $C$, and $D$ represent the set of all things that are simple, straight, lazy, and late, respectively. Write an expression that conveys David Byrne's lyrics in the language of set theory.

*Once you've written up your answer to this problem, take a minute to **type-check** it. As an example, suppose that you have the following answer:*

$$(A \in B) \cap (A \in C)$$

*This expression can't be correct, and here's one way to see this. The expression $A \in B$ is of type Boolean – either $A \in B$ is true or it isn't – and the same is true of $A \in C$. However, the intersection operator $\cap$ can only be applied to sets. The expression therefore contains a type error: it tries to apply an operator that only works on sets to Boolean values.*

*You can type-check this answer in a different way. For example, suppose you came up with this expression:*

$$A \cup F$$

*Here, A and F are sets, so it's perfectly fine to write $A \cup F$, which evaluates to an object of type set. But notice that the statement here asks you to write an expression that says "facts are simple and facts are straight / facts are lazy and facts are late," and that statement is of type Boolean (facts either have these properties or they don't). Therefore, $A \cup F$ can't possibly be an expression with the right meaning, since the type of the expression (set) doesn't match the type of the statement (Boolean).*

*If you're having trouble with this problem, consider working backwards. You know that you need an expression that evaluates to type Boolean. What operations on sets do you have that produce Booleans?*

ii. Suppose you're on an exciting first date. Let $Y$ represent the set of all your hobbies and $D$ represent the set of all your date's hobbies. Write an expression that says that you have a hobby that your date doesn't have.

*As before, type-check your answer! Should your answer represent a set? A Boolean? Something else?*

iii. Let's say that a ***committee*** is a group of people, which we can think of as being represented by the set of people that comprise it. Let's have $S$ represent the set of all students at Stanford and let $F$ represent the set of all faculty at Stanford. Write an expression representing the set of all committees you can make from Stanford students and faculty that contain at least one student and at least one faculty member. You can assume no one is both a student and a faculty member.

*Something to think about: how would you say "all committees made purely of students?" Something else to think about: what is the type of the statement to translate? Is it a Boolean? A set? Neither?*

## Problem Four: Proof Warmups

It's time to write your first proofs! In this problem, we've provided three theorems and partial proofs of those results. We'd like you to fill in the blanks to complete the proofs.

We've provided these partial proofs to help give you a sense of how to structure proofs of different claims. As you move forward in this problem set and start writing out proofs without our scaffolding, feel free to refer back to these partial proofs for guidance.

i. Fill in the blanks to complete this direct proof.

> ***Theorem:*** For any integer $n$, if $n$ is odd, then $n^2$ is odd.
>
> ***Proof:*** _____ odd integer $n$. Since $n$ is odd, there is an integer $k$ where _____. Then we see that
>
> $$\begin{aligned} n^2 &= \text{\_\_\_\_\_} \\ &= \text{\_\_\_\_\_} \\ &= \text{\_\_\_\_\_}. \end{aligned}$$
>
> Therefore, there is an integer $m$ (namely, _____) such that _____, so _____, as required. ■

*The relative sizes of the blanks are not meant as an indicator of how much or how little you need to write in each section. If you'd like to add or remove lines in the chain of equalities, feel free to do so.*

ii. Fill in the blanks to complete this proof by contrapositive.

> ***Theorem:*** For any integers $a$, $b$, and $c$, if $a^2 + b^2 = c^2$, then at least one of $a$, $b$, and $c$ is even.
>
> ***Proof:*** We will prove the contrapositive of this statement, namely, _____.
>
> Choose any integers $a$, $b$, and $c$ where _____. Since $a$, $b$, and $c$ are _____, we know that $a^2$, $b^2$, and $c^2$ are _____.
>
> Because $a^2$ and $b^2$ are _____, there exist integers $p$ and $q$ such that _____. This means that $a^2 + b^2$ = _____ = _____, which means that $a^2 + b^2$ is _____.
>
> However, as mentioned earlier we know that $c^2$ is _____. Therefore, we see that _____, as required. ■

iii. Fill in the blanks to complete this proof by contradiction.

> ***Theorem:*** For any integers $m$ and $n$, if $mn$ is even and $m$ is odd, then $n$ is even.
>
> ***Proof:*** Assume for the sake of contradiction that _____. Since $m$ is _____, we know that there is an integer $k$ where _____. Similarly, since $n$ is _____, there is an integer $r$ where _____. Then we see that
>
> $$\begin{aligned} mn &= \text{\_\_\_\_\_} \\ &= \text{\_\_\_\_\_}, \\ &= \text{\_\_\_\_\_}, \end{aligned}$$
>
> which means that $mn$ is _____, but this is impossible because _____.
>
> We have reached a contradiction, so our assumption must have been wrong. Therefore, if $mn$ is even and $m$ is odd, then $n$ is even. ■

## Problem Five: Modular Arithmetic

Different numbers can yield the same remainder when divided by some number. For example, the numbers 1, 12, 23, and 34 all leave a remainder of one when divided by eleven. To formalize this relationship between numbers, we'll introduce a relation $\equiv_k$ that, intuitively, indicates that two numbers leave the same remainder when divided by $k$. For example, we'd say that $1 \equiv_{11} 12$, since both 1 and 12 leave a remainder of 1 when divided by 11, and that $8 \equiv_3 14$, since both 8 and 14 leave a remainder of 2 when divided by 3.

To be more rigorous, we'll formally define $\equiv_k$. For any integer $k$, define $a \equiv_k b$ as follows:

We say that $a \equiv_k b$ if there exists an integer $q$ such that $a = b + kq$

We'd read the statement "$x \equiv_k y$" aloud as "*$x$ is congruent to $y$ modulo $k$*." For example, $7 \equiv_3 4$, because $7 = 4 + 3 \cdot 1$, and $5 \equiv_4 13$ because $5 = 13 + 4 \cdot (-2)$.

i. Fill in the blanks to complete this proof about modular congruence.

> ***Theorem:*** For any integers $x$ and $y$ and any integer $k$, if $x \equiv_k y$, then $y \equiv_k x$.
>
> ***Proof:*** Let $x$, $y$, and $k$ be arbitrary integers where $x \equiv_k y$. We will prove that $y \equiv_k x$. To do so, we will prove that there is an integer $q$ where _____.
>
> Because $x \equiv_k y$, we know there is an integer $r$ such that _____. Now, let $q$ = _____. Then we see that
>
> $$\begin{aligned} y &= \text{\_\_\_\_\_} \\ &= \text{\_\_\_\_\_} \\ &= \text{\_\_\_\_\_}, \end{aligned}$$
>
> which is what we needed to show. ■

*The relative sizes of the blanks here aren't meant to indicate how much you need to write in each spot. Also, for the blanks in the chain of equalities, feel free to add or remove blanks if you find yourself needing more or less space.*

ii. Prove for any integers $x$, $y$, $z$, and $k$ that if $x \equiv_k y$ and $y \equiv_k z$, we have $x \equiv_k z$.

iii. Prove for any integers $x$ and $k$ that $x \equiv_k x$.

*This proof will feel different from the previous problems because you aren't assuming anything about x or k other than the fact that they're integers. But the basic setup will feel the same: you're searching for some choice of q that makes some equality hold. Follow the general idea from earlier when structuring your proof: state, explicitly, what choice of q you're going to make, then prove that it has the right properties.*

## Problem Six: Proof Critiques, Part One

One of the best ways to improve your proof *writing* is to do some proof *reading*. In doing so, you'll get a much better perspective on why certain stylistic conventions matter, both in the positive sense, ("wow, that's so easy to read!") and in the negative sense ("I can't make heads or tails of this!"). You'll also get practice tracing through mathematical arguments, which will help expand your repertoire of techniques and give you a better sense of what details to focus on in your own reasoning.

We've curated three proofs for you to review. Read these proofs, and for each one, do the following:

- ***Critique its style***. Our Proofwriting Checklist contains specific details to review in any proof. Go through the checklist one item at a time, identifying any style errors and explaining each.
- ***Critique its reasoning.*** Are the arguments given by these proofs correct? If there are logic errors, point them out and demonstrate why they're incorrect. Then, determine whether the theorem being proved is even true in the first place. After all, you can prove anything using faulty logic!
- ***If possible, rewrite the proof.*** You now have a list of issues. Based on what you've found, do one of the following:
  - If the reasoning is correct but the proof has poor style, simply rewrite the proof to improve its style, keeping the core argument intact.
  - If the reasoning is *incorrect* but the statement being proved is still true, write a new proof of the overall theorem. Try to modify the original argument as little as possible.
  - If the reasoning is *incorrect* and the statement being proved isn't even true to begin with, briefly explain why the statement isn't true, though no formal disproof is required.

You should submit both your notes from the initial critique and your newly-revised proofs.

i. Critique this proof about parities (the ***parity*** of a number is whether it's even or odd.)

> ***Theorem:*** The sum of an even integer and an odd integer is odd.
>
> ***Proof:*** This proof will talk about adding together different kinds of numbers. An even integer is an integer that can be written as $2k$ for some integer $k$. Therefore, $m = 2k$. Similarly, an odd integer is one that can be written as $2k+1$ for some integer $k$. So $n = 2k+1$. $m + n = 2k + 2k + 1 = 4k + 1$. Therefore $m + n$ is odd. ■

ii. Critique this proof about natural numbers.

> ***Theorem:*** Every natural number is odd.
>
> ***Proof:*** Assume for the sake of contradiction that every natural number is even. In particular, that would mean that 137 is even. Since $137 = 2 \cdot 68 + 1$ and 68 is a natural number, we see that 137 is odd. We know that there is no integer $n$ where $n$ is both odd and even. However, $n = 137$ is both even and odd. This is impossible. We've reached a contradiction, so our assumption must have been wrong. Therefore, every natural number is odd. ■

iii. Critique this proof about modular arithmetic.

> ***Theorem:*** If $n$ is an integer and $n \equiv_2 0$, then $n \equiv_4 0$.
>
> ***Proof:*** We will prove the contrapositive of this statement and show that if $n$ is an integer where $n \equiv_4 0$, then $n \equiv_2 0$. Since $n \equiv_4 0$, we know $n = 0 + 4q$. This in turn tells us that $n = 2(2q)$, so there is an integer $m$ such that $n = 2m$. Consequently, we see that $n = 0 + 2m$, and so $n \equiv_2 0$, as required. ■

## Problem Seven: Proofs on Sets

The remainder of this problem set focuses on proofs on sets and set theory. As a reminder, generally speaking, proofs on sets generally rely on arguments about specific elements of those sets and how they relate to one another, rather than holistic arguments about how the sets behave "in general."

Read over the Guide to Set Theory Proofs, then fill in the blanks to complete this proof.

> ***Theorem:*** If $A$ and $B$ are sets where $A \subseteq B$, then $B = A \cup B$.
>
> ***Proof:*** Pick any two sets $A$ and $B$ where _____. We will prove that _____. To do so, we will prove that _____ $\subseteq$ _____ and that _____ $\subseteq$ _____.
>
> First, we will prove that _____ $\subseteq$ _____. To do so, consider an arbitrary $x \in B$. We will prove that $x \in$ _____. To see this, note that since _____, by definition of set union we see that _____, as required.
>
> Next, we will prove that _____ $\subseteq$ _____. Pick any $x \in A \cup B$. We'll show that $x \in$ _____. To do so, we will consider two cases:
>
> *Case 1:* $x \in$ _____. Then we already know that $x \in$ _____.
>
> *Case 2:* $x \in$ _____. By our assumption that _____, we see that _____.
>
> In either case, we see that _____. Thus _____ $\subseteq$ _____, as required. ■

*As before, the relative sizes of the blanks do not necessarily tell you how much you need to write.*

## Problem Eight: Proof Critiques, Part Two

Critique the following two proofs by following the same instructions as in Problem Six (apply the Proofwriting Checklist, flag any logic errors, determine whether the theorems are true, and then either re-write the proof or briefly explain why the statement in question isn't true to begin with).

We recommend completing Problem Seven and reading the Guide to Set Theory Proofs before starting this problem.

i. Critique this proof about sets.

> ***Theorem:*** If $A \subseteq B$ and $A \subseteq C$, then $A \subseteq B \cap C$.
>
> ***Proof:*** Since $A \subseteq B$, it means that some group of the elements of $B$ is the set $A$. Since $A \subseteq C$, it means that some group of the elements of $C$ is the set $A$. Therefore, some group of the elements of $B \cap C$ is the set $A$, so $A \subseteq B \cap C$. ■

ii. Critique this proof about sets. Here, the notation $S \subsetneq T$ is read as "$S$ is a ***strict subset*** of $T$" and means that $S \subseteq T$ and $S \neq T$.

> ***Theorem:*** If $A \subsetneq B$ and $A \subsetneq C$, then $A \subsetneq B \cap C$.
>
> ***Proof:*** Since $A \subsetneq B$, it means that some group of the elements of $B$ is the set $A$, and there are some other elements of $B$. Since $A \subsetneq C$, it means that some group of the elements of $C$ is the set $A$, and there are some other elements of $C$. Therefore, some group of the elements of $B \cap C$ is the set $A$, and there are some other elements of $B \cap C$, so $A \subsetneq B \cap C$. ■

## Problem Nine: Disproofs

A proof is an argument that shows that a claim is true. A ***disproof*** is an argument that shows that a claim is *false*. Check the Proofwriting Checklist for more information about disproofs.

i. Fill in the blanks to complete the following disproof.

> ***(False) Claim:*** For all sets $A$, $B$, and $C$, if $A \in B$ and $B \in C$, then $A \in C$.
>
> ***Disproof:*** We will prove that the negation of this claim is true. Specifically, we will prove that _____.
>
> Let $A$ = ____, $B$ = ____, and $C$ = ____. Then _____. ■

*The amount you need to write in each blank is, as before, not necessarily indicated by the size of the blank. Feel free to write as many sentences as you'd like in the long blank at the end.*

ii. Fill in the blanks to complete the following disproof.

> ***(False) Claim:*** There is an odd integer $n$ where $n^2 + 2n$ is even.
>
> ***Disproof:*** We will prove that the negation of this claim is true. Specifically, we will prove that _____.
>
> Let $n$ be an arbitrary odd integer. _____. ■

*The same advice from part (i) applies here – feel free to expand that final blank into as many sentences as you'd like.*

## Problem Ten: Properties of Sets

For each of these statements about sets, decide whether that statement is true or false. Prove each true statement, and disprove each false statement.

i. Prove or disprove: if $A$, $B$, and $C$ are sets where $A - C = B - C$, then $A = B$.

*This is your first example of a "prove or disprove" problem. Your first task is to figure out whether it's true or false, since if it's true you'll want to prove it and if it's false you'll want to disprove it.*

*Here are two strategies for approaching problems like these. First,* ***try out a lot of examples!*** *You'll want to build an intuition for what the symbolic expression above "feels" like in practice. Second, get a sheet of scratch paper and write out both the statement and its negation. One of those statements is true, and your task is to figure out which one it is. Once you have those two statements, think about what you would need to do to prove each of them. In each case, what would you be assuming? What would you need to prove? If you can answer those questions, you can explore both options and seeing which one ends up panning out.*

*Finally, remember to call back to definitions! Regardless of whether you're proving or disproving this, you'll want to refer to the formal definition of set equality.*

ii. Prove or disprove: if $A$ and $B$ are sets and $A \in B$, then $\wp(A) \in \wp(B)$.

*All the advice from part (i) of this problem still applies here. Try out some concrete examples. Look at the Guide to Set Theory Proofs and think about what the statement $\wp(A) \in \wp(B)$ means. Also look back at your coding questions from earlier – can you simplify that expression at all?*

iii. Prove or disprove: if $A$ and $B$ are sets and $\wp(A) \in \wp(B)$, then $A \in B$.

*Notice that this implication essentially is the statement from part (ii) with the antecedent and consequent swapped. We sometimes call this the* ***converse*** *of the implication. Every implication is equivalent to its contrapositive, but not all implications are equivalent to their converses.*

*All the advice from the previous problems applies here. Try concrete examples, explore the definitions, and, more generally, explore, explore, explore!*

*Before you turn in these proofs, read over the Proofwriting Checklist and the Guide to Set Theory Proofs and confirm that your proof meets our style criteria. Here are a few specific things to look for:*

- *Make sure your proofs match the definitions of the relevant terms. To prove $S \subseteq T$, pick an arbitrary $x \in S$, then prove $x \in T$ by making claims about $x$. To prove $S = T$, prove $S \subseteq T$ and $T \subseteq S$.*
- *However, avoid restating definitions in the abstract. For example, rather than writing*

  *"We know that $S \subseteq T$ if every element of $S$ is an element of $T$.*
  *Therefore, since we know that $A \subseteq B$ and $x \in A$, we see that $x \in B$."*

  *instead remove that first sentence and just write something like this:*

  *"Since $x \in A$ and $A \subseteq B$, we see that $x \in B$."*

  *Whoever is reading your proof knows all the relevant definitions. They're more interested in seeing* ***how those definitions interact with one another*** *than* ***what those definitions are****.*
- *Make sure you clearly indicate what each variable means and whether it's chosen arbitrarily or to have a specific value. For example, if you refer to variables like A, B, or C, you should clearly indicate whether they're chosen arbitrarily or refer to specific values.*
- *When talking about an arbitrary set A, it's tempting to list off the elements of A by writing something like $A = \{ x_1, x_2, \ldots, x_n \}$. The problem is that writing $A = \{ x_1, x_2, \ldots, x_n \}$ implicitly says that the set A is finite, since you're saying it only has n elements in it. This is a problem if A is infinite. In fact, if A is infinite, because of Cantor's theorem, you can't necessarily write $A = \{ x_1, x_2, x_3, \ldots \}$, since you might run out of natural numbers with which to name the elements of A!*

## Optional Fun Problems

*On each problem set, we'll provide optional fun problems. These are problems that can be solved purely using the techniques you've learned so far, but which will require more thought than the other problems on the problem set. Each problem, in our opinion, has a beautiful solution that will give you a much deeper understanding of core concepts, so we strongly encourage you to play around with them if you get the chance. Please feel free to work on as many of these problem as you'd like, though please only submit a solution to* ***at most one*** *of the optional fun problems on each problem set. If you submit submissions to both problems, we will pick one to grade Arbitrarily and Capriciously.*

*These optional fun problems have no bearing on your grade. However, if you successfully complete at least one Optional Fun Problem on* ***four*** *or more of the problem sets this quarter, the instructors will send you a certificate attesting to that fact and saying how impressed we are. (These certificates carry no official weight with the university, but are something we could mention in a recommendation letter.)*

*As a matter of course policy, we don't provide any hints on the extra credit problems – after all, they're supposed to be challenge problems! However, we're happy to chat about them after the problem sets come due.*

## Optional Fun Problem One: Hat Colors

You and nine friends are brought into a room. The CS103 TAs go around and place a hat on everyone's head. There are ten different colors of hats – red, orange, yellow, green, blue, magenta, black, white, gray, and gold. It's possible that everyone gets a hat of a different color. It's possible that everyone gets a hat of the same color. In fact, aside from the restriction that each hat is one of the ten colors mentioned earlier, there are no restrictions about how many hats there are of each color.

You and your friends can see what color hats everyone else is wearing, but no one can see their own hat color. The staff will then ask everyone to write down a guess of what color their hat is. We'll collect those guesses and reveal them all at once. You and your friends succeed if at least one of you correctly guesses their own hat color. You and your friends lose if no one correctly guesses their own hat color.

Before we place hats on everyone, we'll allow you and your friends to work out a strategy about how you're going to guess, but once the hats are on no communication of any kind – whether explicit or implicit – is permitted between the group. Devise a strategy that *guarantees* at least one person will correctly guess their own hat color, even if we know what strategy you'll be using. Then, using the mathematical notation and tools you've learned so far in this course, formally prove that your strategy works. For example, here are a few strategies you could choose from that *don't* work:

- Everyone could pick which color they're going to guess before going into the room, and then write down that guess regardless of what color hats they see on everyone else. If we assign hats randomly, then the probability that this works is approximately $1 - e^{-1}$ (take CS109 if you're curious why this is!) However, if we overhear your discussion, we could be mean and deliberately give everyone the wrong hat color, making your probability of success exactly 0.
- Everyone could arrange themselves in a circle, then guess the color of the hat of the person immediately to their left. If we choose hat colors randomly, then there's a decent probability that someone will correctly guess their own hat color. But if we know that this is what you're going to do, we can just give everyone a different hat color and you're guaranteed to lose.

*I first heard this problem from Michelle McGhee, CS major (theory track), storyteller, and Ultimate Frisbee player. Her senior project was a program that generated freestyle rap lyrics.*

## Optional Fun Problem Two: Infinite Deviation

In our first class, we sketched a proof of Cantor's theorem. This proof assumed there was a pairing between the elements of a set $S$ and the subsets of $S$, then constructed a set that was different in at least one position from each of the paired sets.

Show that, no matter how you pair the elements of $\mathbb{N}$ with the subsets of $\mathbb{N}$, there's always at least one set $X \subseteq \mathbb{N}$ that differs in infinitely many positions from each of the paired sets. Justify your answer, but no formal proof is necessary.