Exercise 1: Basic set theory

This exercise is designed to help you get used to the notation and conventions for sets. Consider the following sets:

\[ A = \{1, 2, 3, 4\} \]
\[ B = \{2, 2, 2, 1, 4, 3\} \]
\[ C = \{1, \{2\}, \{\{3, 4\}\}\} \]
\[ D = \{1, 3\} \]
\[ E = \mathbb{N} \]
\[ F = \{\mathbb{N}\} \]

Answer the following questions and briefly justify your answers.

a. Which of the above sets, if any, are equal to one another?

b. Is \(D \in A\)? Is \(D \subseteq A\)?

c. Is \(D \in \mathcal{P}(A)\)? Is \(D \subseteq \mathcal{P}(A)\)?

d. What is \(A \cap C\)? What is \(A \cup C\)? What is \(A - C\)?

e. What is \(|B|\)? What is \(|E|\)? What is \(|F|\)?

f. What is \(E - A\)? Express your answer in set-builder notation.
Exercise 2: Much ado about nothing

It can take practice to get used to the empty set, \( \emptyset \). This problem asks you to think about a few different sets related to \( \emptyset \).

a. Give a set equal to \( \emptyset \cup \{\emptyset\} \).

b. Give a set equal to \( \emptyset \cap \{\emptyset\} \).

c. Give a set equal to \( \{\emptyset\} \cup \\{\{\emptyset\}\} \).

d. Give a set equal to \( \{\emptyset\} \cap \\{\{\emptyset\}\} \).

e. Give a set equal to \( \mathcal{P}(\mathcal{P}(\emptyset)) \).
Exercise 3: Describing the world with set theory

The notation of set theory (e.g., ∪, ∩, ℘, ⊆, ∈, etc.) is a great tool for describing the real world. Consider the Talking Heads song “Crosseyed and Painless”, which contains the lyrics

\begin{center}
Facts are simple and facts are straight. \\
Facts are lazy and facts are late.
\end{center}

Let \( F \) be the set of all facts. Let \( A, B, C, \) and \( D \) be the sets of all things that are simple, straight, lazy, and late, respectively. Write an expression that conveys the lyrics in the language of set theory.

In your answer, don't use (1) plain English, (2) set-builder notation, (3) new variables, or (4) logical notation.

Take the time to \emph{type-check} your answer. For instance, if you wrote \((A \in B) \cap (A \in C)\), it couldn't be correct because the expressions \( A \in B \) and \( A \in C \) are Boolean values – they’re either true or false. However, the intersection operator \( \cap \) can only be applied to sets. Therefore, the expression has a type error.

Suppose, instead, you came up with the expression \( A \cup F \). The arguments to \( \cup \) are both of the right type, but the problem asks you to write an expression equivalent to the lyrics, which are making a claim – a statement that can be true or false. Therefore, \( A \cup F \) can't be an expression with the right meaning since the type of the expression (\emph{set}) doesn't match the type of the statement (\emph{Boolean}).

If you’re having trouble with this problem, consider working backwards. You know you need an expression that evaluates to the type \emph{Boolean}. What operations on sets produce Booleans?
Exercise 4: Set operations in Racket

a. Using Racket, define a function called `intersection` that takes as input two sets, represented as lists. The function should return a list representing the intersection of the two input sets.

b. Similarly, define a `union` function.

*Note:* If the input lists do not have any duplicates, then neither should the output list.

*Guidelines*

Whenever you write a function in Racket:

- Begin by writing its contract – what types of inputs it does it consume and what type of output does it produce? E.g., a function named `friendly-numbers?` that consumes two natural numbers and returns true or false would have the signature

  ```racket
  ;; friendly-numbers? : Natural Natural -> Boolean
  ```

- Next write your test cases using `check-expect` statements. Sometimes I’ll provide test cases, but these won't be exhaustive; you’ll need to think of additional cases.

- Then write the function body based on your test cases. How could you produce the desired output for each desired input?

- When your tests pass, review the code you’ve written. Do you have redundant cases? Should your function actually be two smaller functions? If any parts of the function are mysterious, either rewrite the code to be clearer or, if this isn't possible, write comments to help the reader.

*Submitting*

1. Save your definitions file into an appropriately named directory (e.g., `cmpu145/asmt01`).

2. Submit electronically using the `submit145` command:

   ```
   submit145 asmt01 your-directory-name
   ```

If you’ve typed your answers to exercises 1–3, you can include a PDF of them in the same electronic submission. Alternatively, they can be submitted on paper.