Proving that a decision problem Y is NP-hard requires several steps:[1]

- Choose a decision problem X that you already know is NP-hard (because we told you so in class, or it says so in the textbook).
- ***Reduce X to Y.*** Describe how to transform an instance x of X into an instance y of Y such that x is a YES instance of X if and only if y is a YES instance of Y.

  If it helps, you might imagine you're designing an algorithm to solve X by transforming its input into one that you can feed to a magic black-box algorithm for Y.
- ***Prove*** that your reduction is correct. This always requires two separate steps, which are usually of the following form:
  - ***Prove*** that your algorithm always transforms YES instances of X into YES instances of Y.
  - ***Prove*** that if your transformation algorithm produces a YES instance of Y, then it was given a YES instance of X. Equivalently, prove that your algorithm always transforms NO instances of X into NO instances of Y.

    (If both X and Y are in NP, these proofs usually mean showing existence of a verifiable *certificate* for X's input implies existence of a verifiable certificate for Y's instance and vice versa.)
- Argue that your reduction takes polynomial time. (This is usually trivial.)

This lab in particular asks you to prove NP-hardness of several variants of the *satisfiability* problem **SAT**. Recall that the input to the SAT is a boolean formula in *conjunctive normal form*: a conjunction (AND) of one or more *clauses*, each of which is the disjunction (OR) of one or more *literals*, each of which is either a variable or its negation. The goal of SAT is to determine whether a given CNF formula Φ has a *satisfying assignment*, that is, an assignment of TRUE or FALSE to each variable so that Φ evaluates to TRUE.

We have already seen one special case of SAT this is still NP-hard: **3SAT** requires that every clause in the input formula contains exactly three distinct literals.

---

1. **4SAT** is the special case of SAT where every clause of the input CNF formula Φ contains exactly four distinct literals. Prove that 4SAT is NP-hard. *[Hint: Reduce from 3SAT.]*

2. **BALANCEDSAT** asks whether a given CNF formula Φ has a satisfying assignment where exactly half of the variables are TRUE. Prove that BALANCEDSAT is NP-hard.

3. **EXACT3SAT** (also called ONEIN3SAT) asks whether a given CNF formula Φ, with exactly three distinct literals per clause, has an assignment such that each clause contains *exactly* one TRUE literal. Prove that EXACT3SAT is NP-hard.

---

[1] To prove that decision problem Y is NP-*complete*, you *also* have to argue $Y \in$ NP by describing the polynomial size certificate. Often, Y will be a problem of the form, "is there a *foo* in/for this input?" and the certificate is that *foo*.

Harder problems to think about later:

4. **($\leq$3,$\leq$3)-SAT** is the special case of SAT where each clause of the input CNF formula Φ contains ***at most*** three literals, and each variable appears in at most ***three*** clauses. Prove that ($\leq$3,$\leq$3)-SAT is NP-hard.

5. **(3,$\leq$4)-SAT** is the special case of SAT where each clause of the input CNF formula Φ contains ***exactly*** three distinct literals, and each variable appears in at most ***four*** clauses. Prove that (3,$\leq$4)-SAT is NP-hard.[2] *[Hint: Reduce from ($\leq$3,$\leq$3)-SAT.]*

6. A boolean formula is in ***exclusive-or conjunctive normal form*** (XNCF) if it consists of a conjunction (AND) of one or more clauses, each of which is the ***exclusive-or*** of one or more literals, each of which is either a variable or its negation. Thus, a satisfying assignment for an XCNF formula ensures that each clause contains an odd number of TRUE literals. **XORSAT** asks whether a given XCNF formula has a satisfying assignment.

   Describe a polynomial-time algorithm for XORSAT!

[2] The related problem (3,$\leq$3)-SAT, where each clause in the input formula contains exactly three literals and each variable appears in at most ***three*** clauses, is trivial. All such formulae are satisfiable!