Proving that a language L is undecidable by reduction requires several steps. (These are the essentially the same steps you already use to prove that a problem is NP-hard.)

- Choose a language L' that you already know is undecidable (because we told you so in class). The simplest choice is usually the standard halting language

$$\text{HALT} := \left\{ \langle M, w \rangle \;\middle|\; M \text{ halts on } w \right\}$$

- Describe an algorithm that decides L', using an algorithm that decides L as a black box. Typically your reduction will have the following form:

  Given an arbitrary string x, construct a special string y,
  such that $y \in L$ if and only if $x \in L'$.

  In particular, if $L = \text{HALT}$, your reduction will have the following form:

  Given the encoding $\langle M, w \rangle$ of a Turing machine M and a string w,
  construct a special string y, such that
  $y \in L$ if and only if M halts on input w.

- Prove that your algorithm is correct. This proof almost always requires two separate steps:
  - Prove that if $x \in L'$ then $y \in L$.
  - Prove that if $x \notin L'$ then $y \notin L$.

Very important: Name every object in your proof, and *always* refer to objects by their names. Never *ever* refer to “the Turing machine” or “the algorithm” or “the code” or “the input string” or (gods forbid) “it” or “this”, even in casual conversation, even if you’re “just” explaining your intuition, even when you’re “just” *thinking* about the reduction to yourself.

Prove that the following languages are undecidable.

1. $\text{AcceptIllini} := \left\{ \langle M \rangle \;\middle|\; M \text{ accepts the string } \texttt{ILLINI} \right\}$
2. $\text{AcceptThree} := \left\{ \langle M \rangle \;\middle|\; M \text{ accepts exactly three strings} \right\}$
3. $\text{AcceptPalindrome} := \left\{ \langle M \rangle \;\middle|\; M \text{ accepts at least one palindrome} \right\}$
4. $\text{AcceptOnlyPalindromes} := \left\{ \langle M \rangle \;\middle|\; \text{Every string accepted by } M \text{ is a palindrome} \right\}$

A solution for problem 1 appears on the next page; don’t look at it until you’ve thought a bit about the problem first.

Solution (for problem 1): For the sake of argument, suppose there is an algorithm DecideAcceptIllini that correctly decides the language AcceptIllini. Then we can solve the halting problem as follows:

```
DecideHalt(⟨M, w⟩):
  Encode the following Turing machine M′:
      M′(x):
        ⟨⟨ignore the input string x⟩⟩
        run M on input w
        ⟨⟨ignore the output of M⟩⟩
        return True
  if DecideAcceptIllini(⟨M′⟩)
      return True
  else
      return False
```

We prove this reduction correct as follows:

$\Longrightarrow$ Suppose M halts on input w.

Then M' accepts *every* input string x.

In particular, M' accepts the string `ILLINI`.

So DecideAcceptIllini accepts the encoding $\langle M' \rangle$.

So DecideHalt correctly accepts the encoding $\langle M, w \rangle$.

$\Longleftarrow$ Suppose M does not halt on input w.

Then M' diverges on *every* input string x.

In particular, M' does not accept the string `ILLINI`.

So DecideAcceptIllini rejects the encoding $\langle M' \rangle$.

So DecideHalt correctly rejects the encoding $\langle M, w \rangle$.

In both cases, DecideHalt is correct. But that's impossible, because Halt is undecidable. We conclude that the algorithm DecideAcceptIllini does not exist. ■

As usual for undecidablility proofs, this proof invokes *four* distinct Turing machines:

- The hypothetical algorithm DecideAcceptIllini.
- The new algorithm DecideHalt that we construct in the solution.
- The arbitrary machine M whose encoding is part of the input to DecideHalt.
- The special machine M' whose encoding DecideHalt constructs (from the encoding of M and w) and then passes to DecideAcceptIllini.