Prove that the following languages are undecidable.

1. \( \text{AcceptIllini} := \{ \langle M \rangle \mid M \text{ accepts the string } \text{ILLINI} \} \)

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

```plaintext
DecideHalt(\langle M, w \rangle):
Encode the following Turing machine \( M' \):

\[
M'(x):
\text{run } M \text{ on input } w
\text{return True}
\]

if DecideAcceptIllini(\langle M' \rangle)
return True
else
return False
```

We prove this reduction correct as follows:

\( \Rightarrow \) Suppose \( M \) halts on input \( w \).
Then \( M' \) accepts every input string \( x \).
In particular, \( M' \) accepts the string \( \text{ILLINI} \).
So \( \text{DecideAcceptIllini} \) accepts the encoding \( \langle M' \rangle \).
So \( \text{DecideHalt} \) correctly accepts the encoding \( \langle M, w \rangle \).

\( \Leftarrow \) Suppose \( M \) does not halt on input \( w \).
Then \( M' \) diverges on every input string \( x \).
In particular, \( M' \) does not accept the string \( \text{ILLINI} \).
So \( \text{DecideAcceptIllini} \) rejects the encoding \( \langle M' \rangle \).
So \( \text{DecideHalt} \) correctly rejects the encoding \( \langle M, w \rangle \).

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

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

- The hypothetical algorithm \( \text{DecideAcceptIllini} \).
- The new algorithm \( \text{DecideHalt} \) that we construct in the solution.
- The arbitrary machine \( M \) whose encoding is part of the input to \( \text{DecideHalt} \).
- The special machine \( M' \) whose encoding \( \text{DecideHalt} \) constructs (from the encoding of \( M \) and \( w \)) and then passes to \( \text{DecideAcceptIllini} \).
2. \textit{AcceptThree} := \{ ⟨M⟩ \mid M \text{ accepts exactly three strings} \}

\textbf{Solution:} For the sake of argument, suppose there is an algorithm \textsc{DecideAcceptThree} that correctly decides the language \textit{AcceptThree}. Then we can solve the halting problem as follows:

```
\textsc{DecideHalt}(⟨M, w⟩):

Encode the following Turing machine \(M'\):

\(M'(x):\)
- run \(M\) on input \(w\)
- if \(x = \epsilon\) or \(x = 0\) or \(x = 1\)
  - return \text{True}
- else
  - return \text{False}

if \textsc{DecideAcceptThree}(⟨M'⟩)
  return \text{True}
else
  return \text{False}
```

We prove this reduction correct as follows:

\(\Rightarrow\) Suppose \(M\) halts on input \(w\).
- Then \(M'\) accepts exactly three strings: \(\epsilon, 0, \text{ and } 1\).
- So \textsc{DecideAcceptThree} accepts the encoding \(⟨M'⟩\).
- So \textsc{DecideHalt} correctly accepts the encoding \(⟨M, w⟩\).

\(\Leftarrow\) Suppose \(M\) does not halt on input \(w\).
- Then \(M'\) diverges on every input string \(x\).
- In particular, \(M'\) does not accept exactly three strings (because \(0 \neq 3\)).
- So \textsc{DecideAcceptThree} rejects the encoding \(⟨M'⟩\).
- So \textsc{DecideHalt} correctly rejects the encoding \(⟨M, w⟩\).

In both cases, \textsc{DecideHalt} is correct. But that’s impossible, because \textit{Halt} is undecidable. We conclude that the algorithm \textsc{DecideAcceptThree} does not exist. 
\[\blacksquare\]
3. \textbf{AcceptPalindrome} := \{ \langle M \rangle \mid M \text{ accepts at least one palindrome} \}

\textbf{Solution:} For the sake of argument, suppose there is an algorithm \textsc{DecideAcceptPalindrome} that correctly decides the language \textsc{AcceptPalindrome}. Then we can solve the halting problem as follows:

\begin{verbatim}
\textsc{DecideHalt}(\langle M, w \rangle):  
  Encode the following Turing machine \( M' \):
  \( M'(x): \)
  \begin{tabular}{l}
  run \( M \) on input \( w \) \\
  return True
  \end{tabular}
  \begin{tabular}{l}
  if \textsc{DecideAcceptPalindrome}(\langle M' \rangle) \\
  return True
  \end{tabular}
  else
  return False
\end{verbatim}

We prove this reduction correct as follows:

\( \Rightarrow \) Suppose \( M \) halts on input \( w \).
  Then \( M' \) accepts every input string \( x \).
  In particular, \( M' \) accepts the palindrome \textsc{RACECAR}.
  So \textsc{DecideAcceptPalindrome} accepts the encoding \( \langle M' \rangle \).
  So \textsc{DecideHalt} correctly accepts the encoding \( \langle M, w \rangle \).

\( \Leftarrow \) Suppose \( M \) does not halt on input \( w \).
  Then \( M' \) diverges on every input string \( x \).
  In particular, \( M' \) does not accept any palindromes.
  So \textsc{DecideAcceptPalindrome} rejects the encoding \( \langle M' \rangle \).
  So \textsc{DecideHalt} correctly rejects the encoding \( \langle M, w \rangle \).

In both cases, \textsc{DecideHalt} is correct. But that’s impossible, because \textsc{Halt} is undecidable. We conclude that the algorithm \textsc{DecideAcceptPalindrome} does not exist.

Yes, this is \textbf{exactly} the same proof as for problem 1.

4. \textbf{AcceptReversed} := \{ \langle M \rangle \mid M \text{ accepts } w^R \text{ whenever it accepts } w \}

\textbf{Solution:} HW Problem