Prove that the following languages are undecidable.

1. **AcceptILLINI** := \{ \langle M \rangle \mid M \text{ accepts the string } \text{ILLINI} \}

**Solution:** 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(\langle M, w \rangle):
    Encode the following Turing machine M:
    M'(x):
        run M on input w
        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 **DECIDEAcceptILLINI** accepts the encoding \(\langle M' \rangle\).
So **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 **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. \[\blacksquare\]

As usual for undecidability 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**.
2. \textbf{AcceptThree} := \{ \langle M \rangle \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 \textsc{AcceptThree}. Then we can solve the halting problem as follows:

\textsc{DecideHalt}(\langle M, w \rangle):

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

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

if \textsc{DecideAcceptThree}(\langle M' \rangle)
  return True
else
  return False

We prove this reduction correct as follows:

\( \implies \) Suppose \( M \) halts on input \( w \).
  Then \( M' \) accepts exactly three strings: \( \epsilon \), \( 0 \), and \( 1 \).
  So \textsc{DecideAcceptThree} accepts the encoding \( \langle M' \rangle \).
  So \textsc{DecideHalt} correctly accepts the encoding \( \langle M, w \rangle \).

\( \impliedby \) 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 \( \langle M' \rangle \).
  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{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{center}
\begin{tabular}{|l|}
\hline
\textsc{DecideHalt(}\langle M, w \rangle)\text{):
\end{tabular}
\end{center}

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

\begin{itemize}
\item \( M'(x) \):
\begin{itemize}
\item run \( M \) on input \( w \)
\item return \text{True}
\end{itemize}
\end{itemize}

if \textsc{DecideAcceptPalindrome(}\langle M' \rangle)\text{):
\begin{itemize}
\item return \text{True}
\end{itemize}
else
\begin{itemize}
\item return \text{False}
\end{itemize}

We prove this reduction correct as follows:

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

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

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.  \hfill \blacksquare