Prove that the following languages are undecidable.

See outline of how to solve such problems in the original problem set.

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

**Solution:**

For the sake of argument, suppose there is an algorithm \texttt{DecideAcceptILLINI} that correctly decides the language \texttt{AcceptILLINI}. Then we can solve the halting problem as follows:

\begin{algorithm}
\textbf{DecideHalt}(\langle M, w \rangle):
\begin{itemize}
    \item Encode the following Turing machine \( M' \):
        \begin{algorithm}
            \textbf{M'}(x):
            \begin{itemize}
                \item run \( M \) on input \( w \)
                \item return \texttt{TRUE}
            \end{itemize}
            if \texttt{DecideAcceptILLINI}(\langle M' \rangle)
            \begin{itemize}
                \item return \texttt{TRUE}
            \end{itemize}
        \end{itemize}
        \begin{itemize}
            \item else
            \begin{itemize}
                \item return \texttt{FALSE}
            \end{itemize}
        \end{itemize}
\end{algorithm}
\end{itemize}
\end{algorithm}

We prove this reduction correct as follows:

\begin{itemize}
    \item \( \implies \) Suppose \( M \) halts on input \( w \).
        Then \( M' \) accepts \emph{every} input string \( x \).
        In particular, \( M' \) accepts the string \textit{ILLINI}.
        So \texttt{DecideAcceptILLINI} accepts the encoding \( \langle M' \rangle \).
        So \texttt{DecideHalt} correctly accepts the encoding \( \langle M, w \rangle \).
    \item \( \iff \) Suppose \( M \) does not halt on input \( w \).
        Then \( M' \) diverges on \emph{every} input string \( x \).
        In particular, \( M' \) does not accept the string \textit{ILLINI}.
        So \texttt{DecideAcceptILLINI} rejects the encoding \( \langle M' \rangle \).
        So \texttt{DecideHalt} correctly rejects the encoding \( \langle M, w \rangle \).
\end{itemize}

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

As usual for undecidability proofs, this proof invokes \emph{four} distinct Turing machines:

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

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

\begin{verbatim}
DECIDEHALT(\langle M, w \rangle):
    Encode the following Turing machine \texttt{M'}:
    \texttt{M'}(x):
        run \texttt{M} on input \texttt{w}
        if \( x = \varepsilon \) or \( x = 0 \) or \( x = 1 \)
            return \texttt{TRUE}
        else
            return \texttt{FALSE}
    if \texttt{DecideAcceptThree}(\langle \texttt{M'} \rangle)
        return \texttt{TRUE}
    else
        return \texttt{FALSE}
\end{verbatim}

We prove this reduction correct as follows:

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

\[ \iff \] Suppose \texttt{M} does not halt on input \texttt{w}.
Then \texttt{M'} diverges on every input string \texttt{x}.
In particular, \texttt{M'} does not accept exactly three strings (because \( 0 \neq 3 \)).
So \texttt{DecideAcceptThree} rejects the encoding \( \langle \texttt{M'} \rangle \).
So \texttt{DecideHalt} correctly rejects the encoding \( \langle \texttt{M}, \texttt{w} \rangle \).

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

AcceptPalindrome := \{ \langle M \rangle \mid M \text{ accepts at least one palindrome} \} 

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

\begin{verbatim}
DECIDEHALT(\langle M, w \rangle):
    Encode the following Turing machine \texttt{M'}:
    \texttt{M'}(x):
        run \texttt{M} on input \texttt{w}
        return \texttt{TRUE}
    if \texttt{DecideAcceptPalindrome}(\langle \texttt{M'} \rangle)
        return \texttt{TRUE}
    else
        return \texttt{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 \textit{RACECAR}.
  So \texttt{DecideAcceptPalindrome} accepts the encoding \( \langle M' \rangle \).
  So \texttt{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 \texttt{DecideAcceptPalindrome} rejects the encoding \( \langle M' \rangle \).
  So \texttt{DecideHalt} correctly rejects the encoding \( \langle M, w \rangle \).

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

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