Prove that the following languages are undecidable.

1. $\textsc{AcceptIllini} := \{\langle M\rangle \mid M \text{ accepts the string } \texttt{ILLINI}\}$

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(⟨M, w⟩):
  Encode the following Turing machine M′:
      M′(x):
        run M on input w
        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.

2. AcceptThree $:= \{\langle M\rangle \mid M \text{ accepts exactly three strings}\}$

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

```
DecideHalt(⟨M, w⟩):
  Encode the following Turing machine M′:
      M′(x):
        run M on input w
        if x = ILL or x = UMIN or x = ATI
            return True
        else
            return False
  if DecideAcceptThree(⟨M′⟩)
      return True
  else
      return False
```

We prove this reduction correct as follows:

$\Longrightarrow$ Suppose M halts on input w.
Then M' accepts exactly three strings, namely ILL, UMIN, and ATI.
So DecideAcceptThree 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 exactly three strings (because $0 \neq 3$).
So DecideAcceptThree 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 DecideAcceptThree does not exist. ■

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

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

```
DecideHalt(⟨M, w⟩):
  Encode the following Turing machine M′:
      M′(x):
        run M on input w
        return True
      if DecideAcceptPalindrome(⟨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 palindrome `ILLINILLI`.
So DecideAcceptPalindrome 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 any palindromes.
So DecideAcceptPalindrome 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 DecideAcceptPalindrome does not exist.

Yes, this is ***exactly*** the same proof as for problem 1. ■

4. $\textsc{AcceptOnlyPalindromes} := \{\langle M\rangle \mid \text{Every string accepted by } M \text{ is a palindrome}\}$

Solution: For the sake of argument, suppose there is an algorithm DecideAcceptOnlyPalindromes that correctly decides the language AcceptOnlyPalindromes. Then we can solve the halting problem as follows:

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

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 non-palindrome `ILLINI`.
So DecideAcceptOnlyPalindromes rejects 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.
Thus, vacuously, every string that M' accepts is a palindrome.
So DecideAcceptOnlyPalindromes accepts 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 DecideAcceptOnlyPalindromes does not exist. ■