Rice’s Theorem. Let \( \mathcal{L} \) be any set of languages that satisfies the following conditions:

- There is a Turing machine \( Y \) such that \( \text{Accept}(Y) \in \mathcal{L} \).
- There is a Turing machine \( N \) such that \( \text{Accept}(N) \notin \mathcal{L} \).

The language \( \text{AcceptIn}(\mathcal{L}) := \{ \langle M \rangle \mid \text{Accept}(M) \in \mathcal{L} \} \) is undecidable.

You may find the following Turing machines useful:

- \( M_{\text{Accept}} \) accepts every input.
- \( M_{\text{Reject}} \) rejects every input.
- \( M_{\text{Hang}} \) infinite-loops on every input.

Equivalently, these machines implement the following algorithms:

\[
\text{IsString}(w): \quad \text{return } \text{True} \quad \text{IsGiraffe}(w): \quad \text{return } \text{False} \quad \text{Duuuuuude}(w): \quad \text{loop forever}
\]

Prove that the following languages are undecidable using Rice’s Theorem:

1. \( \text{AcceptRegular} := \{ \langle M \rangle \mid \text{Accept}(M) \text{ is regular} \} \)
2. \( \text{AcceptILLINI} := \{ \langle M \rangle \mid M \text{ accepts the string ILLINI} \} \)
3. \( \text{AcceptPalindrome} := \{ \langle M \rangle \mid M \text{ accepts at least one palindrome} \} \)
4. \( \text{AcceptThree} := \{ \langle M \rangle \mid M \text{ accepts exactly three strings} \} \)
5. \( \text{AcceptUndecidable} := \{ \langle M \rangle \mid \text{Accept}(M) \text{ is undecidable} \} \)

**Solution:** Undecidability proofs for all of these languages appear in the undecidability lecture notes. ■
To think about later. Which of the following languages are undecidable? How would you prove that? We know several ways to prove undecidability:

- **Diagonalization:** Assume the language is decidable, and derive an algorithm with self-contradictory behavior.
- **Reduction:** Assume the language is decidable, and derive an algorithm for a known undecidable language, like \textsc{Halt} or \textsc{SelfReject} or \textsc{NeverAccept}.
- **Rice’s Theorem:** Find an appropriate family of languages \( L \), a machine \( Y \) that accepts a language in \( L \), and a machine \( N \) that does not accept a language in \( L \).
- **Closure:** If two languages \( L \) and \( L' \) are decidable, then the languages \( L \cap L' \), \( L \cup L' \), \( L \setminus L' \), and \( L^* \) are all decidable, too.

1. \( \text{Accept}\{\varepsilon\} := \{(M) \mid M \text{ accepts only the string } \varepsilon \}; \text{ that is, } \text{Accept}(M) = \{\varepsilon\} \)

   **Solution: Undecidable by Rice’s theorem.** Let \( \mathcal{L} = \{\{\varepsilon\}\} \) — the set containing one language, which contains one string, which is empty. Let \textsc{IsEmpty} be a Turing machine that implements the following algorithm:

   \[
   \textbf{IsEmpty}(w):
   \begin{align*}
   \text{if } w &= \varepsilon \\
   &\text{return True} \\
   \text{else} &\text{return False}
   \end{align*}
   \]

   Clearly \( \text{Accept}(\textsc{IsEmpty}) = \{\varepsilon\} \in \mathcal{L} \). On the other hand, \( \text{Accept}(\textsc{IsGiraffe}) = \text{Accept}(\textsc{Duuuuude}) = \emptyset \notin \mathcal{L} \). ■

2. \( \text{Accept}\{\emptyset\} := \{(M) \mid M \text{ does not accept any strings} \}; \text{ that is, } \text{Accept}(M) = \emptyset \)

   **Solution: Undecidable by Rice’s theorem.** Let \( \mathcal{L} = \{\emptyset\} \) — the set containing one language, which contains no strings. We immediately have \( \text{Accept}(\textsc{IsGiraffe}) = \emptyset \in \mathcal{L} \) but \( \text{Accept}(\textsc{IsString}) = \Sigma^* \notin \mathcal{L} \). ■

3. \( \text{Accept} \emptyset := \{(M) \mid \text{Accept}(M) \text{ is not an acceptable language} \} \)

   **Solution: Trivially decidable.** For any Turing machine \( M \) the language \( \text{Accept}(M) \) is acceptable by definition. Thus, \( \text{Accept} \emptyset = \emptyset \) is correctly decided by \textsc{IsGiraffe}. ■

4. \( \text{Accept} = \text{Reject} := \{(M) \mid \text{Accept}(M) = \text{Reject}(M) \} \)

   **Solution: Undecidable by definition-chasing.** \( \text{Accept}(M) = \text{Reject}(M) \) if and only if \( M \) diverges on every input string. Thus, \( \text{Accept} = \text{Reject} = \text{NeverHalt} \), which is proved undecidable in the notes. ■
**Solution: Undecidable by reduction.** But in case we don’t have the notes handy, we can prove \( \text{NEVERHALT} \) undecidable by reduction from \( \text{HALT} \). Suppose for the sake of argument that the machine \( NH \) decides \( \text{NEVERHALT} \). Consider the following algorithm:

\[
H(M, w):
\]
Write the encoding \( \langle Wtf \rangle \) of the following machine:

\[
Wtf(x):
\]
throw \( x \) out the window
run \( H \) on \( w \)

if \( NH(\langle Wtf \rangle) \)
return \( \text{FALSE} \)
else
return \( \text{TRUE} \)

If \( M \) halts on \( w \), then \( Wtf \) always halts, which implies that \( NH \) must reject \( \langle Wtf \rangle \), which means \( H \) accepts \( \langle M \rangle, w \). On the other hand, if \( M \) hangs on \( w \), then \( Wtf \) never halts, which implies that \( NH \) accepts \( \langle Wtf \rangle \), which implies that \( H \) rejects \( \langle M \rangle, w \). So algorithm \( H \) correctly solves the halting problem, which is impossible.

**Solution: Undecidable by diagonalization.** We can prove \( \text{NEVERHALT} \) undecidable by a direct diagonalization argument. Suppose for the sake of argument that the machine \( NH \) decides \( \text{NEVERHALT} \). Consider the following algorithm:

\[
NH'(M):
\]
Write the encoding \( \langle Wtf \rangle \) of the following machine:

\[
Wtf(x):
\]
throw \( x \) out the window
run \( M \) on \( \langle M \rangle \)

if \( NH \) rejects \( \langle Wtf \rangle \)
loop forever

\( \langle \text{always returns} \rangle \)

Now consider the behavior of \( NH' \) given its own encoding \( \langle NH' \rangle \) as input.

- If \( NH' \) halts on input \( \langle NH' \rangle \), then \( Wtf \) always halts, so \( NH \) rejects \( \langle Wtf \rangle \), so \( NH' \) loops forever.
- If \( NH' \) hangs on input \( \langle NH' \rangle \), then \( Wtf \) always hangs, so \( NH \) accepts \( \langle Wtf \rangle \), so \( NH' \) halts.

In both cases, we obtain a contradiction.
5. $\text{Accept} \neq \text{Reject} := \{ \langle M \rangle \mid \text{Accept}(M) \neq \text{Reject}(M) \}$

**Solution: Undecidable by closure.** $\text{Accept}(M) \neq \text{Reject}(M)$ if and only if $M$ halts on at least one input string. Thus, $\text{NeverHalt} = \text{Encodings} \setminus \text{Accept} \neq \text{Reject}$, where $\text{Encodings}$ is the language of all machine encodings. $\text{Encodings}$ is decidable, but $\text{NeverHalt}$ is not (see above). Thus, $\text{Accept} \neq \text{Reject}$ cannot be decidable. ■

**Solution: Undecidable by reduction.** We can prove that $\text{SometimesHalt}$ (a more mnemonic name for $\text{Accept} \neq \text{Reject}$) is undecidable by reduction from $\text{Halt}$. Suppose for the sake of argument that the machine $SH$ decides $\text{SometimesHalt}$. Consider the following algorithm:

```
H(\langle M \rangle, w):
  Write the encoding \langle Wtf \rangle of the following machine:
  \begin{align*}
  \text{Wtf}(x): & \quad \text{throw } x \text{ out the window} \\
  \text{run } H \text{ on } w \\
  \end{align*}
  \text{if } SH(\langle Wtf \rangle) \\
  \quad \text{return } \text{True} \\
  \text{else} \\
  \quad \text{return } \text{False}
```

If $M$ halts on $w$, then $\text{Wtf}$ always halts, which implies that $SH$ must accept $\langle \text{Wtf} \rangle$, which means $H$ accepts $\langle M \rangle, w$. On the other hand, if $M$ hangs on $w$, then $\text{Wtf}$ never halts, which implies that $SH$ rejects $\langle \text{Wtf} \rangle$, which implies that $H$ rejects $\langle M \rangle, w$. So algorithm $H$ correctly solves the halting problem, which is impossible. ■

**Solution: Undecidable by diagonalization.** We can also prove $\text{SometimesHalt}$ undecidable by a direct diagonalization argument. Suppose for the sake of argument that the machine $SH$ decides $\text{SometimesHalt}$. Consider the following algorithm:

```
SH'(\langle M \rangle):
  Write the encoding \langle Wtf \rangle of the following machine:
  \begin{align*}
  \text{Wtf}(x): & \quad \text{throw } x \text{ out the window} \\
  \text{run } M \text{ on } \langle M \rangle \\
  \end{align*}
  \text{if } SH \text{ accepts } \langle \text{Wtf} \rangle  \\
  \quad \text{\langle always returns} \rangle \\
  \text{loop forever}
```

Now consider the behavior of $SH'$ given its own encoding $\langle SH' \rangle$ as input.

- If $SH'$ halts on input $\langle SH' \rangle$, then $\text{Wtf}$ always halts, so $SH$ accepts $\langle \text{Wtf} \rangle$, so $SH'$ loops forever.
- If $SH'$ hangs on input $\langle SH' \rangle$, then $\text{Wtf}$ always hangs, so $SH$ rejects $\langle \text{Wtf} \rangle$, so $SH'$ halts.

In both cases, we obtain a contradiction. ■
6. $\text{Accept} \cup \text{Reject} := \{ (M) \mid \text{Accept}(M) \cup \text{Reject}(M) = \Sigma^* \}$

**Solution: Undecidable by definition-chasing.** $\text{Accept}(M) \cup \text{Reject}(M) = \Sigma^*$ if and only if $M$ halts on every input string. Thus, $\text{Accept} \cup \text{Reject} = \text{AlwaysHalt} = \text{NeverDiverge}$, which is proved undecidable in the notes.

**Solution: Undecidable by reduction.** We can prove $\text{AlwaysHalt}$ undecidable by reduction from $\text{Halt}$. Suppose for the sake of argument that the machine $AH$ decides $\text{AlwaysHalt}$. Consider the following algorithm:

\[ \text{H}((M), w): \]
Write the encoding $\langle Wtf \rangle$ of the following machine:
\[
\begin{align*}
Wtf(x): \\
\text{run } H \text{ on } w \\
\text{if } AH(\langle Wtf \rangle) \\
\text{return True} \\
\text{else} \\
\text{return False}
\end{align*}
\]

If $M$ halts on $w$, then $Wtf$ never diverges, which implies that $AH$ must accept $\langle Wtf \rangle$, which means $H$ accepts $\langle M \rangle, w$. On the other hand, if $M$ hangs on $w$, then $Wtf$ always diverges, which implies that $AH$ must reject $\langle Wtf \rangle$, which implies that $H$ rejects $\langle M \rangle, w$. So algorithm $H$ correctly solves the halting problem, which is impossible.

Yes, this is exactly the same proof as the previous problem.

**Solution: Undecidable by diagonalization.** We can also prove $\text{AlwaysHalt}$ undecidable by a direct diagonalization argument. Suppose for the sake of argument that the machine $AH$ decides $\text{AlwaysHalt}$. Consider the following algorithm:

\[ \text{AH}'((M)): \]
Write the encoding $\langle Wtf \rangle$ of the following machine:
\[
\begin{align*}
Wtf(x): \\
\text{throw } x \text{ out the window} \\
\text{run } M \text{ on } \langle M \rangle \\
\text{if } AH \text{ accepts } \langle Wtf \rangle, \langle \text{always returns} \rangle \\
\text{loop forever}
\end{align*}
\]

Now consider the behavior of $NH'$ given its own encoding $\langle NH' \rangle$ as input.

- If $AH'$ halts on input $\langle AH' \rangle$, then $Wtf$ always halts, so $AH$ accepts $\langle Wtf \rangle$, so $AH'$ loops forever.
- If $AH'$ hangs on input $\langle AH' \rangle$, then $Wtf$ always hangs, so $AH$ rejects $\langle Wtf \rangle$, so $AH'$ halts.

In both cases, we obtain a contradiction.

Yes, this is exactly the same proof as the previous problem.