Exercise: (Section 3.3.2 from textbook)
Consider the following languages over \( \Sigma = \{0,1\} \):

\[
L_e = \{ \langle E(M) \rangle \mid L(M) = \emptyset \}
\]

\[
L_{\bar{e}} = \{ \langle E(M) \rangle \mid L(M) \neq \emptyset \}
\]

\( L_e \) is the set of all strings that encode TMs that accept the empty language.

\( L_{\bar{e}} \) is the complement of \( L_e \).

**Claim 1:** \( L_{\bar{e}} \) is R.E.

**Proof:** construct NTM \( N \) for \( L_{\bar{e}} \)

(and then convert \( N \) to an ordinary TM.)

\( N \) works as follows: on input \( E(M) \):

1) guess a string \( w \in \Sigma^* \)
2) simulate \( M \) on \( w \) (like a UTM)
3) accept \( E(M) \) if \( M \) accepts \( w \)

\[\begin{array}{ccc}
\text{guessed } w & \rightarrow & \text{U} \\
E(M) & \rightarrow & \text{yes} \\
& \rightarrow & \text{yes} \\
& \rightarrow & \text{N}
\end{array}\]

We have \( E(M) \in L_{\bar{e}}(N) \iff \exists w \text{ s.t. } \langle M, w \rangle \in L(U) \iff \exists w \text{ s.t. } w \in L(M) \iff E(M) \in L_{\bar{e}} \)
Claim 2: \( L_{ne} \) is non-recursive.

Proof: by reduction from \( L_m \) to \( L_{ne} \).

Reduction \( R \) is a function computable by a halting T.M. with input: instance \( \langle M, w \rangle \) of \( L_m \)

output: instance \( \varepsilon(M') \) of \( L_{ne} \)

end set: \( \langle M, w \rangle \in L_m \iff \varepsilon(M') \in L_{ne} \)

Description of \( M' \):
- \( M' \) ignores completely its own input string \( X \)
- instead, it replaces its input by the string \( \langle M, w \rangle \) and runs \( M \) on \( w \) (see (*) below)
- if \( M \) accepts \( w \), then \( M' \) accepts \( X \)
- if \( M \) never halts on \( w \) or rejects \( w \)
  then \( M' \) also never halts or rejects \( X \)

Note: if \( w \in \mathcal{L}(M) \to \mathcal{L}(M') = \Sigma^* \)
if \( w \notin \mathcal{L}(M) \to \mathcal{L}(M') = \emptyset \)

hence \( \langle M, w \rangle \in L_m \iff \varepsilon(M') \in L_{ne} \)

We can construct a halting T.M. \( M_{R} \) that, given \( \langle M, w \rangle \) as input, reconstructs \( \varepsilon(M') \) for an \( M' \) that behaves as above.

q.e.d.

(*) \( M' \) has the following form:

\[
\begin{array}{c}
\text{Input } X \\
\text{Write } w \text{ on the tape} \\
\text{Go to the beginning of } w \text{ and} \\
\text{Runs } M \text{ on } w
\end{array}
\]

To sum up, we have that \( L_{ne} \) is RE but non-recursive.

Hence \( L_{ne} \) must be non-RE.
Exercise 3.2.1

The halting problem, \( L_{\text{halt}} \), is the set \( \langle M, w \rangle \) s.t.
\( M \) halts on \( w \) (with or without accepting) is RE.
but not recursive.

To show RE, we construct a T.M. \( H \) s.t.
\( L(H) = L_H = \{ \langle M, w \rangle \mid M \text{ halts on } w \} \)

To show that \( L_H \) is not recursive, we assume by contradiction
it is RE, and derive that \( L_m \) is recursive.

By contradiction, let \( H \) be an algorithm for \( L_H \), and \( V \) a procedure for \( L_m \)

\( A_m \) would be an algorithm for \( L_m \).
Contradiction.
Let $L$ be R.E. and $\overline{L}$ be non-R.E.

Consider $L' = \{0w | w \in L\} \cup \{1w | w \notin L\}$.

What do we know about $L'$ and $\overline{L'}$?

We show that $L'$ is non-R.E.

Suppose by contradiction that we have a procedure $M_L$ for $L'$.

Then we can construct a procedure $M_{\overline{L}}$ for $\overline{L}$ as follows:
- on input $w$, $M_{\overline{L}}$ changes the input to $1w$ and simulates $M_L$.
- if $M_L$ accepts $1w$, then $w \in L$, and $M_{\overline{L}}$ accepts.
- if $M_L$ does not terminate or terminates and answers no, then $w \notin L$, and $M_{\overline{L}}$ does not terminate or terminates and answers no.

$\Rightarrow M_{\overline{L}}$ would accept exactly $\overline{L}$. Contradiction.

$\overline{L'} = \{0w | w \in L\} \cup \{1w | w \in L\} \cup \{\epsilon\}$

Reasoning as for $L'$, we get that $\overline{L'}$ is non-R.E.
Fl, the complement of the halting problem, i.e., the set of pairs \( \langle M, w \rangle \) such that \( M \) on input \( w \) does not halt, is non-R.E.

Proof: By reduction from \( L_1 \), which is non-R.E.

Idea: we show how to convert any TM \( M \) into another TM \( M_h \) such: \( M_h \) halts on \( w \) iff \( M \) accepts \( w \).

Construction:
1) Ensure that \( M_h \) does not halt unless \( M \) accepts.
   - Add to the states of \( M \) a new loop state \( q \), with 
     \[ \delta(q, x) = (q, x, y) \] for all \( x \in \Gamma \)
   - For each \( \delta(q, y) \) that is undefined and \( q \in F \), 
     add \( \delta(q, y) = (p, y, f) \)

2) Ensure that, if \( M \) accepts, then \( M_h \) halts.
   - Make \( \delta(q, x) \) undefined for all \( q \in F \) and \( x \in \Gamma \)

3) The other moves of \( M_h \) are as those of \( M \).

Q.E.D.