**Problem 1.** Prove that any property on *pairs* of RE sets is undecidable.

Let $P_p : (L_1, L_2) \to \{T, F\}$ be a property on pairs of RE sets. We map $P_p$ to $P_s$, a property on single sets, and use Rice’s Theorem to show that $P_p$ is undecidable.

We define our map as $(L_1, L_2) \mapsto \{w_1 \$ w_2 \mid w_1 \in L_1, w_2 \in L_2\}$, and define $P_s$ as follows:

$$P_s(X) = \begin{cases} T & \text{if } P_p(L_1, L_2) = T \\ F & \text{if } \exists a \in X, a \text{ is not of the form } w_1 \$ w_2 \\ F & \text{otherwise} \end{cases}$$

Then, if $P_p$ is non-trivial, then there is a pair $(L_1, L_2)$ such that

$$P_p((L_1, L_2)) = T.$$  

This means that there exists

$$L' = \{w_1 \$ w_2 \mid w_1 \in L_1, w_2 \in L_2\}$$

such that $P_s(L') = T$, and by the same reasoning, there must be a set $L_3$ which is false under $P_s$. Hence, $P_s$ is also non-trivial.

Hence any non-trivial property on pairs of RE sets corresponds to a property on single RE sets. By Rice’s Theorem we know $P_s$ is undecidable, so any non-trivial property on pairs of RE sets is also undecidable.
#2 (Exercise 9.3.4)

a) Let \( L_2 = \{ \langle M \rangle : |L(M)| \geq 2 \} \). Here, \( \langle M \rangle \) denotes a string encoding of a Turing machine \( M \). \( L_2 \) is RE (i.e., recognizable).

We can build a Turing machine \( T \) that accepts (recognizes) \( L_2 \). For \( k = 1, 2, 3, \ldots \), \( T \) simulates \( M \) on \( w_1 \) to \( w_k \) for at most \( k \) steps, and see if \( M \) accepts. (This is called dovetailing.) \( T \) keeps a count of how many strings has been accepted by \( M \), and accepts if two has been found.

b) Let \( L_\infty = \{ \langle M \rangle : |L(M)| = \infty \} \). \( L_\infty \) is not RE (i.e., unrecognizable).

We can also show that if \( L_\infty \) is recognizable, then the halting problem is decidable. The proof is similar to that of Rice's theorem, with a slight twist.

Let \( R \) be a recognizer for \( L_\infty \). We now build a decider \( H \) for the halting problem. \( H(\langle M, x \rangle) \) accepts if \( M \) halts on input \( x \), and rejects otherwise.

\( H(\langle M, x \rangle) \):

- Construct a machine \( M' \) such that \( M' \) does the following.

  \( M'(w) \):
  - Simulate \( M(x) \) for at most \( k = |w| \) steps.
  - If \( M \) did not halt within \( k \) steps, then accept \( w \).

- Simulate \( M(x) \) and \( R(\langle M' \rangle) \) in “parallel”. That is, simulate \( M(x) \) for one step, then simulate \( R(\langle M' \rangle) \) for one step, and so on.

- Accept if \( M(x) \) ever halts.

- Reject if \( R(\langle M' \rangle) \) ever accepts.

The crux of this solution is the construction of \( M' \). Observe that if \( M(x) \) does not halt, then \( M' \) will accept all strings, so \( L(M') = \Sigma^* \), which is infinite. Otherwise, if \( M(x) \) does halt after \( k \) steps, then \( L(M') = \{ w : |w| < k \} \), which is a finite set!

By running \( M(x) \) and \( R(\langle M' \rangle) \) in parallel, \( H \) can always decide whether \( M \) halts on input \( x \). Here’s how: Suppose it does halt. Then the simulation of \( M(x) \) will eventually halt and we accept. Suppose \( M(x) \) does not halt. Then \( L(M') = \Sigma^* \), so
our recognizer $R$ for $L_\infty$ is required to halt at some point and accept $\langle M' \rangle$. At this point, $H$ can halt and reject.

c) Let $L_{\text{CFL}} = \{ \langle M \rangle : L(M) \text{ is context-free} \}$. $L_{\text{CFL}}$ is not RE.

Same proof as part b), except we build $M'$ as follows.

$M'(w)$:

- Simulate $M(x)$.
- Accept $w$ if it’s of the form $a^n b^n c^n$.

If $M(x)$ does not halt, then $L(M') = \emptyset$, which is context-free. Otherwise, if $M(x)$ does halt, then $L(M') = \{a^n b^n c^n\}$, which is not context-free.

d) Let $L_R = \{ \langle M \rangle : L(M) = L(M)^R \}$. $L_R$ is not RE.

Same format as part b). Construct $M'$ as follows.

$M'(w)$:

- Simulate $M(x)$.
- Accept $w$ if it’s of the form $a^n b^n$.

If $M(x)$ does not halt, then $L(M') = \emptyset$, and thus $L(M') = L(M')^R$ trivially. Otherwise, $L(M') = \{a^n b^n\}$, so $L(M') \neq L(M')^R$. 
1. (a) Let \( M \) be such a (deterministic) machine. The key insight is that while the machine is in the input area, it behaves exactly like a two-way finite automaton. Assume without loss of generality that anytime \( M \) wants to accept, it moves past the right end of the input string before entering the accept state. The proof that \( L(M) \) is regular is otherwise identical to the proof that two-way finite automata accept only regular sets; see a previous homework set. Briefly, each string \( x \) determines a table

\[
T_x : (Q \cup \{\bullet\}) \rightarrow (Q \cup \{\bot\}),
\]

where \( T_x(\bullet) \) is the state \( M \) is in the first time it emerges to the right of \( x \), or \( \bot \) if it never emerges; and if \( M \) enters the input from the right in state \( p \), then \( T_x(p) \) is the state \( M \) is in the first time it emerges again to the right of \( x \), or \( \bot \) if it never emerges. We define \( x \equiv y \) if \( T_x = T_y \). Note that if \( x \equiv y \), then given \( xz \) as an input, if we replace the string \( x \) by \( y \), then the computation of the Turing machine in the blank tape will remain exactly the same. In fact, the acceptance/rejection will also remain the same, i.e. \( xz \) is accepted by \( M \) iff \( yz \) is accepted by \( M \). Thus the acceptance of \( xz \) is completely determined by \( T_x \). Since the relation \( \equiv \) is of finite index (there are only finitely many possible tables), by the Myhill-Nerode Theorem, \( L(M) \) is regular.

(b) Given \( M \# x \), construct a machine \( M' \) that does the following on any input \( y \):

- moves immediately to the right of \( y \) and lays down an endmarker \( \vdash \);
- writes \( x \) on the tape to the right of the \( \vdash \) (\( M' \) has \( x \) hard-wired in its finite control);
- runs \( M \) on \( x \) (\( M' \) also has a description of \( M \) hard-wired in its finite control);
- accepts iff \( M \) halts on \( x \).

Then \( M' \) never overwrites the input string. If \( M \) halts on \( x \), then \( M' \) accepts its input \( y \), and if \( M \) does not halt on \( x \), then \( M' \) loops on \( y \). Moreover, \( M' \) does the same thing for all \( y \). Therefore if \( M \) halts on \( x \), then \( L(M') = \Sigma^* \), and if \( M \) does not halt on \( x \), then \( L(M') = \emptyset \). Note that both of these are regular sets. If we could decide which of these two regular sets \( M' \) accepts, then we would have decided whether \( M \) accepts \( x \).

Now if it were possible to effectively produce from \( M' \) an equivalent finite automaton \( F \), then we could decide whether \( L(M) = \Sigma^* \) or \( \emptyset \), since \( L(F) = L(M) \) and these questions are decidable for finite automata. This would allow us to decide the halting problem.
b) Let $L_\emptyset = \{ \langle M \rangle : M \text{ halts on no input} \}$. $L_\emptyset$ is not RE.

The proof is exactly the same as that for 9.3.4(b) (problem 2), except we construct $M'$ as follows.

$M'(w)$:

- Simulate $M(x)$.
- Accept $w$.

If $M$ does not halt on $x$, then $M'$ never halts regardless of its own input $w$. If $M$ does halt on $x$ at some point, then $M'$ halts on all inputs.

c) This is the complement of $L_\emptyset$. We know it can’t be recursive, since if $\overline{L_\emptyset}$ is recursive, then so is $L_\emptyset$. But it is RE.

We can build a recognizer $T$ for $\overline{L_\emptyset}$. For $k = 1, 2, 3, \ldots$, $T$ simulates $M$ on $w_1$ to $w_k$ for at most $k$ steps. If $M$ ever halts on a simulation, $T$ accepts.