Problem 1

Prove that

\[ E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \} \]

is unrecognizable.

**Proof by contradiction:** Assume that \( E_{TM} \) is recognizable. That is, there exists a TM, \( R_{E}\langle M \rangle \), that can recognize \( E_{TM} \).

We have already shown that \( E_{TM} \) is undecidable. This suffices to show as well that \( E_{TM} \) is undecidable, where \( E_{TM} \) is defined as:

\[ E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts at least one string} \} \]

We will now show that \( E_{TM} \) is recognizable by constructing a TM \( R_{E} \) that can recognize it. \( R_{E} \) will function as follows:

1.) Enact a method of enumerating all possible input strings. For example, strings might be ordered by length, starting with the empty string, and then alphabetically.

2.) Simulate \( M \) on the first input string for one time step.

3.) Simulate \( M \) on the first input string for two time steps. Simulate \( M \) on the second input string for one time step.

4.) Simulate \( M \) on the first input string for three time steps. Simulate \( M \) on the second input string for two time steps. Simulate \( M \) on the third input string for one time step.

5.) Continue this process until one of these simulations accepts, at which point ACCEPT.

**Case 1:** \( \langle M \rangle \in E_{TM} \) In this case \( M \) accepts at least one string. Using the above dovetailing method, we can ensure that if such an accept state exists, it will be reached eventually by the recognizer. When this happens the recognizer will accept.

Based on our initial assumption that \( E_{TM} \) is recognizable and our derivation that \( E_{TM} \) is recognizable, we can now build a functional decider for the language \( E_{TM} \). However we have already proved that \( E_{TM} \) is undecidable and that this decider cannot exist. Thus we can conclude that our initial assumption was false and that \( E_{TM} \) is unrecognizable.
Problem 2
Prove that
\[ L = \{ \langle M \rangle | M \text{ accepts one of the following strings } \text{"101"}, \text{"001"}, \text{"111"}\} \]
is undecidable. Is \( L \) recognizable? Is \( \overline{L} \) recognizable?

Proof by reduction: \( 101_{\text{TM}} \) to \( L \)

Initial Assumption: There exists a TM, \( D_L\langle M \rangle \), that can decide \( L \).

We will now build a decider for \( 101_{\text{TM}} \) that takes input \( \langle M \rangle \) and uses \( D_L\langle M \rangle \) to determine if \( M \) accepts input “101”. This decider, \( D_{101}\langle M \rangle \), will function as follows:

1.) Use \( M \) to construct a new machine \( M' \) that immediately rejects strings “001” and “111” and then defers control to the original \( M \).

2.) Run \( D_L \) on \( M' \)

3.) If \( D_L \) accepts, then ACCEPT

4.) If \( D_L \) rejects, then REJECT

Case 1: \( \langle M \rangle \in 101_{\text{TM}} \) In this case \( M \) accepts input “101”, which means that \( M' \) will also accept input “101”. Thus \( D_L\langle M' \rangle \) will accept causing \( D_{101}\langle M \rangle \) to accept.

Case 2: \( \langle M \rangle \notin 101_{\text{TM}} \) This means that \( M \) does not accept input “101”. We know that \( M' \) automatically rejects inputs “001”, and “111”, so if \( M \) does not accept input “101”, then \( M' \) does not accept any of the inputs “101”, “001”, and “111”. This will cause \( D_L\langle M' \rangle \) to reject, in turn causing \( D_{101}\langle M \rangle \) to reject.

L is recognizable. We can build a recognizer using the same dovetailing approach used in Problem 1. Because L is undecidable and recognizable we can conclude that \( \overline{L} \) is unrecognizable.