Problem 1

Consider the following language:

\[ L = \{ \langle M \rangle \mid M \text{ accepts input } \varepsilon \} \]

We wish to prove that \( L \) is undecidable. Recall that we know that

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ accepts } w \} \]

is undecidable.

Candidate Proof: Assume there exists a decider \( A \) for \( A_{TM} \). Then we construct the following decider \( D \) for \( L \): \( D \) takes input \( \langle M \rangle \) and runs \( A \) on \( \langle M, \varepsilon \rangle \), then outputs whatever \( A \) returns. Clearly, \( D \) will halt because it is only running \( A \), which is a decider. \( D \) also clearly decides \( L \), because it only returns true when \( M \) accepts \( \varepsilon \), and false otherwise. However, this is a contradiction, since \( A_{TM} \) is not decidable, so \( L \) must not be decidable.

1. What is wrong with the proof given above? Explain.
2. How would you correctly prove that \( L \) is undecidable?

Problem 2

For each of the following languages, explain why it is decidable or undecidable. If it is undecidable, explain why you can or cannot use Rice’s Theorem.
a. \( L = \{\langle M \rangle \mid M \text{ is a TM and } \text{tr}ololol \in L(M)\} \).

b. \( L = \{\langle M, w, s \rangle \mid M \text{ at some point writes symbol } s \text{ on the tape given input } w\} \)

c. \( L = \{\langle M \rangle \mid |L(M)| \leq 9000\} \)

\textbf{Problem 3}

Show that the following languages are decidable.

a. \( L = \{\langle M \rangle \mid M \text{ is a DFA and } L(M) = \emptyset\} \)

b. \( L = \{\langle M, N \rangle \mid M \text{ and } N \text{ are DFAs with } L(M) = L(N)\} \)

c. \( L = \{\langle M, N \rangle \mid M, N \text{ are DFAs such that } L(M) \subseteq L(N)\} \)

d. \( L = \{\langle M \rangle \mid M \text{ is a DFA that accepts } w \text{ whenever it accepts the reverse of } w, \text{ or } w^R\} \)

\textbf{Hint}: For some problems, you might want to consider operations on regular languages such as taking the union of two languages or the complement of a language.
The following questions are lab problems.

Lab Problem 1

In the following problem, $M$ denotes a Turing machine. Determine whether or not each of the following languages is decidable. Justify your answer.

a. $L = \{\langle M, w \rangle \mid \text{on input } w, \text{there is a state of } M \text{ that is never visited, excluding the accept and reject states} \}$

b. $L = \{\langle M, w \rangle \mid \text{on input } w, M\text{'s head reaches the end of } w; \text{ that is, } M \text{ reads every symbol in } w \}$

c. $L = \{\langle M, w \rangle \mid \text{on input } w, \text{at each step, } M \text{ only writes the symbol already on the tape (leaving the tape unchanged) or writes the blank symbol onto the tape} \}$

Lab Problem 2

A Turing Machine is **self-terminating** if it halts when given its own description as input.

Our goal will be to prove that the language $L_{NST} = \{\langle M \rangle \mid M \text{ is not self-terminating} \}$ is undecidable.

a. Show whether or not you are able to apply Rice’s Theorem.

b. What language should you reduce from? Why?

c. Show the reduction from the language in part (b) to $L_{NST}$.

**Hint:** Create a mapping and show that it maps elements from the language in part (b) to $L_{NST}$, and also maps their complements.

Put all of these steps together in a formal proof, including why the Turing machine you gave in part (c) decides the language in part (b).

Lab Problem 3

So far we have learned about decidability and recognizability. In this problem, we will learn about function uncomputability.
We define a function \( f \) from \( \{0,1\}^* \to \{0,1\}^* \) to be **computable** if there exists a TM that on input \( x \) halts and accepts with the string \( f(x) \) on its tape.

We say a function is **uncomputable** if it is not computable.

We now define LOLCOMPLEXITY of a string \( x \) to be the minimum number of states of a Turing Machine that outputs \( x \) on input the empty string \( \varepsilon \). Note: the smallest possible Turing Machine does not have to be unique.

Prove by contradiction that LOLCOMPLEXITY is uncomputable. You may assume that there exist strings with arbitrarily large LOLCOMPLEXITY.

**Hint:** Think about the expression “the smallest positive integer that cannot be described in fewer than 100 words”. Why can’t such an integer exist?