Problem 1

Consider the following two languages. For each language, determine whether you can use Rice’s Theorem to prove it is undecidable. If so, use Rice’s Theorem to prove it is undecidable. If not, explain why you cannot use Rice’s Theorem, and prove it is undecidable without using Rice’s Theorem.

a. $L_{yuca} = \{(M) \mid |L(M)| \geq 1\}$

b. $L_{eek} = \{(M) \mid \langle M \rangle \in L(M)\}$

Problem 2

Rudolph Bega is trying to determine whether playing hard-to-get with Ginger, the girl he matched with on JicaMate, is a good strategy. He knows it was a good idea to avoid using his full name—after all, now someone’s interested in him. Rudy’s idea of playing hard-to-get is to not send messages to the people he matches with; i.e. he simply sends the empty string. He thinks he has a good understanding of the decision-making algorithms that Ginger uses when determining the attractiveness of a potential match based off the JicaMate messages she receives. In other words, Rudy is trying to design a Turing machine that can take in the description of Ginger’s decision-making Turing machine, $M$, and determine whether $M$ will accept playing hard-to-get.
Consider the following language:

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

Rudy suspects that \( L \) is undecidable. He knows that \( A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w \} \) is undecidable, and tries to write a proof to justify his suspicions.

**Rudy’s 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. \( D \) will halt because it is only running \( A \), which is a decider. \( D \) also 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.

a. Rudy looks back at his proof (which he made late last night), and has a lot of regrets. What is wrong with his proof? Explain.

b. How would Rudy correctly prove that \( L \) is undecidable?

**Problem 3**

A Turing machine is colloquially known as a *Potato* if, on each input \( x \), it either halts or, if it does not halt, it eventually reaches a configuration it has previously visited. A language is *Potatable* if there exists a Potato that recognizes it. Show that \( L \) is Potatable if and only if it is decidable.
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.

**Hint:** Problem 3 may help with parts (c) and (d).

a. $L_a = \{ \langle M, N, P \rangle \mid M, N \text{ and } P \text{ are DFAs with } L(M) = L(N) \circ L(P) \}$

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

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

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

Lab Problem 2

Rudy Bega and Ginger O’Nion are trying to decide whether they are compatible enough to meet face-to-face. They only want to meet if they have enough traits in common. Neither of them has much time to spend on romance, as Ginger is busy maintaining the O’Nion’s mangelwurzel empire and Rudy is still working on carrot-potato hybrids. Luckily, Rudy and Ginger have Turing machines, $M_R$ and $M_G$, whose languages are binary string representations of their interests. Rudy and Ginger will only meet if they share at least $k$ interests. To help figure out whether they will click, they want to construct a Turing machine for the following language:

$$L = \{ \langle M_R, M_G, k \rangle \mid |L(M_R) \cap L(M_G)| \geq k \}$$

Is this language Turing-recognizable? If so, is it decidable? Will Rudy and Ginger finally meet, or will their love be crushed under the unforgiving weight of an impossible computation (as it so often is)? Prove your answers.

**Hint:** You may want to use nondeterminism or an enumerator.
Lab Problem 3

So far we have learned about decidability and recognizability. In this problem, we will learn about function incomputability.

We define a function $f$ from $\{0,1\}^* \rightarrow \{0,1\}^*$ to be computable if there exists a TM that on every 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 the TaroComplexity 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 that the smallest possible Turing machine does not have to be unique.

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

**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?