Problem 1. Draw a Turing machine that decides the language
\[ A = \{0^k1^m0^k \mid 0 \leq k < m\}. \]
You may assume \( \Sigma = \{0,1\} \).

Problem 2. Recall that a \textit{computable function} \( f : \Sigma^* \to \Sigma^* \) is one for which there is a
Turing machine \( M \) such that on all inputs \( \omega \in \Sigma^* \), \( M(\omega) \) halts with just \( f(\omega) \) on its tape.

Show that the function \( f(x) = 2x \) is computable by drawing a Turing machine for it. Here, \( x \) is a binary number (with its most significant bit given first) and \( f \)'s output in binary should be left on the tape (in the same format). You may assume that the input alphabet is \( \Sigma = \{0,1\} \).

Problem 3. Prove that if a language is decidable, then there is an enumerator which enumerates that language in lexicographical order.

Problem 4. A \textit{linear bounded automaton} (LBA) is a restricted Turing machine. Instead
of having a tape of unlimited length, it has a tape of length linear in its input. In many
ways, this is a more realistic model of computation than Turing machines as our physical
computers work with limited memory. The class of languages which LBAs recognize are
the \textit{context-sensitive} languages. They fall between the context-free and the decidable
languages in terms of computational complexity.

An equivalent definition of an LBA is to impose a further restriction. Instead of having
a linear length tape, the input is given between a left end marker and a right end marker.
The Turing machine must not move past these end markers and it must not overwrite
them. It is otherwise allowed to modify the tape where the input is given.

Define the analogous set to \( A_{TM} \) for linear bounded automata as
\[ A_{LBA} = \{\langle M, \omega \rangle \mid M \text{ is an LBA and } M(\omega) \text{ accepts}\}. \]
Prove that \( A_{LBA} \) is decidable.

Problem 5. For both of the following languages, prove it is undecidable using Rice’s
Theorem or show that Rice’s Theorem does not apply. You may assume that \( \Sigma = \{0,1\} \)
for each problem.
a) \( B = \{\langle M \rangle \mid M \text{ is a TM and } 10 \in L(M)\} \)
b) \( C = \{\langle M \rangle \mid M \text{ is a TM, } L(M) = 0^*, \text{ and } M \text{ visits at most 3 distinct states}\} \)
Problem 6. Prove that $A_{TM} \not\leq_m E_{TM}$.

Problem 7. Show that if $L \in RE$ and $L \leq_m L'$, then $L \in DEC$.

Problem 8. Show that if $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$.

Problem 9. Consider the following language.

$$ODD_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ contains no even length strings} \}$$

Prove that $ODD_{TM} \notin RE$ via a mapping reduction and $ODD_{TM} \in \text{co–RE}$.

Problem 10. Recall that for a function to be computable, there must be a Turing machine which computes it. This applies to functions whose inputs and outputs are natural numbers as well since there is a (computable) bijection between $\Sigma^*$ and $\mathbb{N}$. This allows us to state some theorems more naturally.

The set of all computable functions $f : \mathbb{N} \to \mathbb{N}$ is countable. However, the set of all functions $f : \mathbb{N} \to \mathbb{N}$ is uncountable. Consequentially, almost all such functions are not computable. This is the same argument we made to show that almost all languages are not in $RE$ or $\text{co–RE}$. Beyond this, there are many other strange properties of uncomputable functions that may surprise you.

Prove that there exists a function which grows faster than any computable function. In other words, show that there is a function $f : \mathbb{N} \to \mathbb{N}$ such that for any computable function $g : \mathbb{N} \to \mathbb{N}$ there exists an $N \in \mathbb{N}$ such that for all $n \geq N$, $f(n) > g(n)$.

More succinctly, show $\exists f : \mathbb{N} \to \mathbb{N} : \forall g : \mathbb{N} \to \mathbb{N}$ computable \( \forall^\infty n, f(n) > g(n) \).

(Hint: Approach this similarly to dovetailing by overcoming one additional computable function at a time)