Problem 1. Draw an (ordinary) Turing machine that decides the language

\[ P = \{ \omega \in \Sigma^* \mid \omega = \omega^R \} \]

Here, \( \Sigma = \{0, 1\} \).

Problem 2. Give a Turing machine that decides the language

\[ P = \{1^p \in \Sigma^* \mid p \text{ is prime} \} \]

Here, \( \Sigma = \{1\} \). You may draw the Turing machine or give an algorithm for it, but your algorithm should describe what is happening on the tape(s).

(Hint: A 3-tape Turing machine and the Sieve of Eratosthenes will make your life easier)

Problem 3. Consider the language

\[ S = \{ \langle M \rangle \mid 101 \in L(M) \} \]

a) Show that \( S \in RE \).

b) Show that \( S \notin co-RE \).

Problem 4. So far we have looked at Turing machines which verify their language by accepting or rejecting their inputs. This allows them to tackle decision problems where languages are encoded (sometimes awkwardly) into strings. For example, the following is one encoding of addition into a decision problem.

\[ ADD_D = \{ x + y = z \mid x, y, z \in 0 \mid 1 \mathbb{Z}_2^* \text{ and the binary values of } x \text{ and } y \text{ sum to } z \} \]

When we write algorithms for practical purposes, we are usually more interested in function problems. Instead of answering yes/no to verify if a string belongs to a language, we instead transform an input into an output. That is our Turing machine reads its input and halts with nothing but its output on its tape. The functional problem equivalent to \( ADD_D \) is to transform inputs \( x \) and \( y \) into their sum \( x + y \).

Draw a Turing machine that, on input \( \omega \in 0 \mid 1 \mathbb{Z}_2^* \), adds 1 to \( \omega \) and then halts with nothing else on the tape. To be clear, \( \omega \) is a binary number and its most significant bit appears to the left of all subsequent bits (i.e. it appears first).

Problem 5. Draw a Turing machine that, on input \( \omega \in 0 \mid 1 \mathbb{Z}_2^* \), multiplies \( \omega \) by 2 and then halts with nothing else on the tape. To be clear, \( \omega \) is a binary number and its most significant bit appears to the left of all subsequent bits (i.e. it appears first).