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

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

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

Solution 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)

Solution 2.

Algorithm 1: SieveOfEratosthenes(ω)

- Scan ω on tape 1
  - If ω ≠ 1³², reject
  - Return tape 1’s head to its first cell

- On the tape 2, place two 1 symbols
  - Return tape 2’s head to its first cell

(loop:) Scan right across ω on tape 1 until □
  - Each time tape 1’s head moves right, move tape 2’s head right
  - When tape 2 reaches □, mark tape 1’s current tape cell
  - Afterward, return tape 2’s head to its first cell

- Place a new 1 at the end of tape 2’s non-blank content
  - If tape 2’s non-blank content is longer than ω, go to finish
  - Return both tape heaps to their first cells

  Go to loop

(finish:) Scan tape 1 to get to the last non-blank symbol
  - If this symbol is marked, reject
  - Otherwise, accept

This algorithm works by marking off (1-indexed) indices of ω that are multiples of the length of the string on the second tape. It first hits all of the multiples of 2, then 3, then 4, and so on. If ω’s length is not prime, then it has some factor p. When the second tape’s length is p, it will mark the last symbol of ω, causing the algorithm to reject. Otherwise, the last symbol of ω is never marked, and the algorithm accepts.

Problem 3. Consider the language

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

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

b) Show that \( S \notin co–RE \).

Solution 3.

Algorithm 2: N = On input \( \langle M \rangle \)

- Run \( M(101) \)
  - If \( M \) accepted, accept
  - Reject

a) Clearly, \( N \) accepts \( \langle M \rangle \) if and only if \( M \) accepts the string 101. Thus \( L(N) = S \), hence \( S \in RE \).
To show that $S \notin co\text{-}RE$, it suffices to show that $S \notin DEC$ since we know $S \in RE$. To do so, we will use Rice’s Theorem.

First, the Turing machine that accepts everything belongs to $S$ but the Turing machine that rejects everything does not. As such, $S$ is nontrivial.

Second, consider two Turing machines $A$ and $B$ such that $L(A) = L(B)$. Then $\langle A \rangle \in S$ iff 101 $\in L(A)$ iff 101 $\in L(B)$ iff $\langle B \rangle \in S$. So $S$ is a property of the Turing machine’s language.

Since $S$ has both of these properties, Rice’s Theorem says that $S \notin DEC$, hence $S \notin co\text{-}RE$.

**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).

**Solution 4.**
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).

Solution 5.