1. A **Moore machine** is a variant of a finite-state automaton that produces output; Moore machines are sometimes called finite-state *transducers*. For purposes of this problem, a Moore machine formally consists of six components:

- A finite set $\Sigma$ called the input alphabet
- A finite set $\Gamma$ called the output alphabet
- A finite set $Q$ whose elements are called states
- A start state $s \in Q$
- A transition function $\delta : Q \times \Sigma \rightarrow Q$
- An output function $\omega : Q \rightarrow \Gamma$

More intuitively, a Moore machine is a graph with a special start vertex, where every node (state) has one outgoing edge labeled with each symbol from the input alphabet, and each node (state) is additionally labeled with a symbol from the output alphabet.

The Moore machine reads an input string $w \in \Sigma^*$ one symbol at a time. For each symbol, the machine changes its state according to the transition function $\delta$, and then outputs the symbol $\omega(q)$, where $q$ is the new state. Formally, we recursively define a *transducer* function $\omega^* : \Sigma^* \times Q \rightarrow \Gamma^*$ as follows:

$$\omega^*(w, q) = \begin{cases} 
\epsilon & \text{if } w = \epsilon \\
\omega(\delta(a, q)) \cdot \omega^*(x, \delta(a, q)) & \text{if } w = ax
\end{cases}$$

Given input string $w \in \Sigma^*$, the machine outputs the string $\omega^*(w, s) \in \Gamma^*$. The **output language** $L^\omega(M)$ of a Moore machine $M$ is the set of all strings that the machine can output:

$$L^\omega(M) := \{\omega^*(w, s) : w \in \Sigma^*\}$$

(a) Let $M$ be an arbitrary Moore machine. Prove that $L^\omega(M)$ is a regular language.

(b) Let $M$ be an arbitrary Moore machine whose input alphabet $\Sigma$ and output alphabet $\Gamma$ are identical. Prove that the language

$$L^= (M) = \{w \in \Sigma^* : w = \omega^*(w, s)\}$$

is regular. $L^= (M)$ consists of all strings $w$ such that $M$ outputs $w$ when given input $w$; these are also called *fixed points* for the transducer function $\omega^*$.

*Hint: These problems are easier than they look!*
2. Prove that the following languages are not regular.
   
   (a) \( \{ w \in \{0 + 1\}^* \mid |\#(0, w) - \#(1, w)| < 5 \} \)
   
   (b) Strings in \((0 + 1)^*\) in which the substrings \(00\) and \(11\) appear the same number of times.
   
   (c) \( \{ 0^m 1^n \mid n/m \text{ is an integer} \} \)

3. Let \( L \) be an arbitrary regular language.
   
   (a) Prove that the language \( \text{palin}(L) := \{ w \mid ww^R \in L \} \) is also regular.
   
   (b) Prove that the language \( \text{drome}(L) := \{ w \mid w^R w \in L \} \) is also regular.
Solved problem

4. Let \( L \) be an arbitrary regular language. Prove that the language \( \text{half}(L) := \{ w \mid ww \in L \} \) is also regular.

**Solution:** Let \( M = (\Sigma, Q, s, A, \delta) \) be an arbitrary DFA that accepts \( L \). We define a new NFA \( M' = (\Sigma, Q', s', A', \delta') \) with \( \epsilon \)-transitions that accepts \( \text{half}(L) \), as follows:

\[
Q' = (Q \times Q \times Q) \cup \{s'\}
\]
\[
s' \text{ is an explicit state in } Q'
\]
\[
A' = \{(h, h, q) \mid h \in Q \text{ and } q \in A\}
\]
\[
\delta'(s', \epsilon) = \{(s, h, h) \mid h \in Q\}
\]
\[
\delta'(h, a, q) = \{(\delta(p, a), h, \delta(q, a))\}
\]

\( M' \) reads its input string \( w \) and simulates \( M \) reading the input string \( ww \). Specifically, \( M' \) simultaneously simulates two copies of \( M \), one reading the left half of \( ww \) starting at the usual start state \( s \), and the other reading the right half of \( ww \) starting at some intermediate state \( h \).

- The new start state \( s' \) non-deterministically guesses the “halfway” state \( h = \delta^*(s, w) \) without reading any input; this is the only non-determinism in \( M' \).
- State \((p, h, q)\) means the following:
  - The left copy of \( M \) (which started at state \( s \)) is now in state \( p \).
  - The initial guess for the halfway state is \( h \).
  - The right copy of \( M \) (which started at state \( h \)) is now in state \( q \).
- \( M' \) accepts if and only if the left copy of \( M \) ends at state \( h \) (so the initial non-deterministic guess \( h = \delta^*(s, w) \) was correct) and the right copy of \( M \) ends in an accepting state.

\[ \blacksquare \]