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

**Solution:** Let $M = (\Sigma, \Gamma, Q, s, \delta, \omega)$ be the given Moore machine. We construct an NFA $M' = (\Sigma', Q', s', A', \delta')$ that accepts $L^\circ(M)$ as follows. First we define the input alphabet and various state sets:

$$
\Sigma' = \Gamma, \quad Q' = Q, \quad s' = s, \quad A' = Q.
$$

The transition function $\delta'$ is defined as follows, for all $q \in Q$ and $b \in \Gamma$:

$$
\delta'(q, b) := \{ \delta(q, a) \mid a \in \Sigma \text{ and } \omega(\delta(q, a)) = b \}.
$$

Less formally, we build $M'$ from $M$ by replacing every transition $p \xrightarrow{a} q$ with $p \xrightarrow{\omega(q)} q$, and then letting every state accept.

Whenever $M'$ reads a symbol $b \in \Gamma$ while in state $q$, it non-deterministically guesses a symbol $a \in \Sigma$ such that $\omega(\delta(q, a)) = b$ and transitions to state $\delta(q, a)$. If there is no such symbol, the current execution thread fails.

Each state $q$ in $M'$ indicates that $M'$ has just read the output string $\omega^*(s, w)$, for some input string $w \in \Sigma^*$ such that $\delta^*(s, w) = q$.

**Rubric:** This is enough for full credit.

For example, in the figure below, for each Moore machine $M$ on the left, we would construct the corresponding NFA $M'$ on the right. In each Moore machine, the input symbols are indicated in red on the edges/transitions, and the output symbols are indicated in blue on the vertices/states.

We can informally argue the correctness of our construction as follows. A walk in an NFA or a Moore machine is a sequence of transitions (that is, either a single state, or a transition followed by a walk).

An accepting walk in an NFA is any walk from the start state to any accepting state. The transition string of an accepting walk is the concatenation of the symbols labeling
each transition. An NFA accepts a string $y$ if and only if there is an accepting walk whose transition string is $y$.

Similarly, the output string of a walk in a Moore machine is the concatenation of the output symbols of the states, ignoring the beginning state. A string $y$ is in the output language of a Moore machine $M$ if and only if there is a walk in $M$ that starts at $s$ and whose output string is $y$.

Now consider our NFA $M'$. Accepting walks in $M'$ starts at $s' = s$ and can end at any state. Every transition in $M'$ is also a transition in $M$ and vice versa, so every walk in $M$ is also a walk in $M'$ and vice versa. The transition string of any walk in $M$ is equal to the output string of the same walk in $M'$. We conclude that $M'$ accepts a string $y$ if and only if $M'$ can output the string $y$.

If we really have to, we can formally prove correctness by tedious inductive definition-chasing. Here we go:

**Lemma 1.** For all states $p, q \in Q$ and every string $x \in \Gamma^*$, we have $q \in (\delta')^*(p, x)$ if and only if there is a string $w \in \Sigma^*$ such that $\delta^*(p, w) = q$ and $\omega^*(p, w) = x$.

**Proof:** Let $x$ be an arbitrary string in $\Gamma^*$, and let $p$ and $q$ be arbitrary states in $Q$. Assume, for every state $r$ and every string $y \in \Gamma^*$ that shorter than $x$, that we have $q \in (\delta')^*(r, x)$ if and only if there is a string $v \in \Sigma^*$ such that $\delta^*(r, v) = q$ and $\omega^*(r, v) = y$. There are two cases to consider:

If $x = \epsilon$, then by definition, $q \in (\delta')^*(p, x)$ if and only if $p = q$. Similarly by definition, $\delta^*(p, w) = q$ and $\omega^*(p, \epsilon) = \epsilon$.

On the other hand, if $x = by$ for some symbol $b \in \Gamma$ and string $y \in \Gamma^*$, then

$q \in (\delta')^*(p, x)$
\[\iff q \in (\delta')^*(r, y) \quad \text{for some } r \in \delta'(s, b)
\[\iff q \in (\delta')^*(\delta(p, a), y) \quad \text{for some } a \in \Sigma \text{ such that } \omega(\delta(p, a)) = b
\[\iff \delta^*(\delta(p, a), v) = q \text{ and } \omega^*(\delta(p, a), v) = y
\quad \text{for some } a \in \Sigma \text{ and } v \in \Sigma^* \text{ such that } \omega(\delta(p, a)) = b
\[\iff \delta^*(p, av) = q \text{ and } \omega^*(p, av) = by \quad \text{for some } a \in \Sigma \text{ and } v \in \Sigma^*
\[\iff \delta^*(p, w) = q \text{ and } \omega^*(p, w) = x \quad \text{for some } w \in \Sigma^*$

Here the first equivalence is by definition of $(\delta')^*$, the second equivalence is by definition of $\delta'$; the third equivalence follows from the induction hypothesis; the fourth equivalence is by definition of $\delta^*$ and $\omega^*$; and the fifth equivalence follows from setting $w = av$. □

The correctness of our construction now follows from Lemma 1 by setting $p = s$. ■
Rubric: 5 points =
+ 1 for a formal, complete, and unambiguous description of a DFA or NFA, including the input alphabet $\Sigma'$
  - No points for the rest of the problem if this is missing.
+ 3 for a correct NFA
  - −1 for a single mistake in the description (for example a typo)
+ 1 for a brief English justification. We explicitly do not want a formal proof of correctness, but we do want one or two sentences explaining how the NFA works.

We are deliberately deviating from the standard DFA/NFA rubric here.
(b) Let $M$ be an arbitrary Moore machine whose input alphabet $\Sigma$ and output alphabet $\Gamma$ are identical. Prove that the language $L^\omega(M) = \{w \in \Sigma^* \mid w = \omega^*(s, w)\}$ is regular.

**Solution:** Let $M = (\Sigma, \Sigma, Q, s, \delta, \omega)$ be the given Moore machine. We construct a DFA $M' = (\Sigma', Q', s', A', \delta')$ that accepts $L^\omega(M)$ as follows:

$\Sigma' = \Sigma$

$Q' = Q \cup \{\text{fail}\}$

$s' = s$

$A' = Q$

\[
\delta'(q, a) = \begin{cases} 
\delta(q, a) & \text{if } \omega(\delta(q, a)) = a \\
\text{fail} & \text{otherwise}
\end{cases}
\]

for all $q \in Q$ and $a \in \Sigma$

\[
\delta'(\text{fail}, a) = \text{fail}
\]

for all $a \in \Sigma$

Less formally, we build $M'$ from $M$ by redirecting every transition $p \xrightarrow{a} q$ where $\omega(q) \neq a$ to a new fail state, and then letting every original state accept.

Whenever $M'$ reads a symbol $a \in \Sigma$ while in state $q \in Q$, it either transitions to state $\delta(q, a)$ or fails, depending on whether $\omega(\delta(q, a)) = a$.

Each state $q$ in $M'$ indicates that $M'$ has just read a string $w$ such that $\delta^*(s, w) = q$ and $\omega^*(s, w) = w$.

**Rubric:** This is enough for full credit.

For example, in the figure below, for each Moore machine $M$ on the left, we would construct the corresponding NFA $M'$ on the right. In each Moore machine, the input symbols are indicated in red on the edges/transitions, and the output symbols are indicated in blue on the vertices/states. The first and third NFAs have no transitions out of their start states, which means they reject every non-empty input; in those two cases we have $L^\omega(M) = \{\epsilon\}$.
Rubric: 5 points =
+ 1 for a formal, complete, and unambiguous description of a DFA or NFA
  – No points for the rest of the problem if this is missing.
+ 3 for a correct NFA
  – −1 for a single mistake in the description (for example a typo).
+ 1 for a brief English justification. We explicitly do not want a formal proof of correctness, but we do want one or two sentences explaining how the NFA works.
2. Prove that the following languages are not regular.

(a) \( \{ w \in (\theta + 1)^* \mid |\theta(w) - |(1, w)| < 5 \} \)

Solution: Let \( F \) be the infinite language \((\theta \theta \theta \theta \theta)^*\).

Let \( x \) and \( y \) be arbitrary strings in \( F \).
Then \( x = \theta^i \) and \( y = \theta^j \) for some non-negative integers \( i \neq j \).
Let \( z = \theta^i \).
Then \( xz = \theta^i \theta^i \in L \), because \( |\theta(xz) - |(1, xz)| = |5i - 5i| = 0 < 5 \).
And \( yz = \theta^j \theta^i \not\in L \), because \( |\theta(yz) - |(1, yz)| = |5j - 5i| = 5|j - i| \geq 5 \).
We conclude that \( F \) is a fooling set for \( L \).
Because \( F \) is infinite, \( L \) cannot be regular.

(b) Strings in \((\theta + 1)^*\) in which the substrings \( \theta \theta \) and \( 11 \) appear the same number of times.

Solution: Let \( F \) be the infinite language \( \theta \theta^* \), the set of all non-empty strings of \( \theta \)s.

Let \( x \) and \( y \) be arbitrary strings in \( F \).
Then \( x = \theta^i \) and \( y = \theta^j \) for some positive integers \( i \neq j \).
Let \( z = \theta^i \).
Then \( xz = \theta^i \theta^i \in L \), because \( \theta \theta \) and \( 11 \) each appear exactly \( i - 1 \) times.
And \( yz = \theta^j \theta^i \not\in L \), because \( \theta \theta \) appears \( i - 1 \) times but \( 11 \) appears \( j - 1 \) times.
We conclude that \( F \) is a fooling set for \( L \).
Because \( F \) is infinite, \( L \) cannot be regular.

Solution: Let \( F \) be the infinite language \((\theta \theta \theta 1)^*\).

Let \( x \) and \( y \) be arbitrary strings in \( F \).
Then \( x = (\theta \theta \theta 1)^i \) and \( y = (\theta \theta \theta 1)^j \) for some non-negative integers \( i \neq j \).
Let \( z = (\theta \theta \theta 1)^j \).
Then \( xz = (\theta \theta \theta 1)^i(\theta \theta \theta 1)^j \in L \), because \( \theta \theta \) and \( 11 \) each appears exactly \( i \) times.
And \( yz = (\theta \theta \theta 1)^i(\theta \theta \theta 1)^j \not\in L \), because \( \theta \theta \) appears \( i \) times but \( 11 \) appears \( j \) times.
We conclude that \( F \) is a fooling set for \( L \).
Because \( F \) is infinite, \( L \) cannot be regular.

(c) \( \{ \theta^n 10^n \mid n/m \text{ is an integer} \} \)

Solution: Let \( F = \{ \theta^n 1 \mid n \text{ is prime} \} \). (Note that \( F \) is not a regular language.)

Let \( x \) and \( y \) be arbitrary strings in \( F \).
Then \( x = \theta^n 1 \) and \( y = \theta^q 1 \), for some prime numbers \( p \neq q \).
Without loss of generality, assume \( p > q \). (Otherwise, swap \( x \) and \( y \).)
Let \( z = \theta^p \).
Then \( xz = \theta^p 10^p \in L \), because \( p/p = 1 \) is an integer.
And \( yz = \theta^q 10^p \not\in L \), because \( p \) is prime and \( 1 < q < p \), so \( p/q \) is not an integer.
We conclude that \( F \) is a fooling set for \( L \).
Because \( F \) is infinite, \( L \) cannot be regular.
Solution: Let $F = \emptyset \cup \{1\}$.

Let $x$ and $y$ be arbitrary strings in $F$.

Then $x = \emptyset^i 1$ and $y = \emptyset^j 1$, for some positive integers $i$ and $j$.

Without loss of generality, assume $i < j$. (Otherwise, swap $x$ and $y$.)

Let $z = \emptyset^i$.

Then $xz = \emptyset^i 1 \emptyset^i \in L$, because $i/i = 1$ is an integer.

And $yz = \emptyset^j 1 \emptyset^i \notin L$, because $0 < i/j < 1$, so $i/j$ is not an integer.

We conclude that $F$ is a fooling set for $L$.

Because $F$ is infinite, $L$ cannot be regular.

Rubric: 10 points = 3 for each part as follows:

• 1 point for the fooling set:
  + $\frac{1}{2}$ for explicitly describing the proposed fooling set $F$.
  + $\frac{1}{2}$ if the proposed set $F$ is actually a fooling set.
  − No credit for the proof if the proposed set is not a fooling set.
  − No credit for the problem if the proposed set is finite.

• 2 points for the proof:
  + $\frac{1}{2}$ for correctly considering arbitrary strings $x$ and $y$
    − No credit for the proof unless both $x$ and $y$ are always in $F$.
    − No credit for the proof unless both $x$ and $y$ can be any string in $F$.
  + $\frac{1}{2}$ for correctly stating a suffix $z$ that distinguishes $x$ and $y$.
  + $\frac{1}{2}$ for proving either $xz \in L$ or $yz \in L$.
  + $\frac{1}{2}$ for proving either $yz \notin L$ or $xz \notin L$, respectively.

• $\frac{1}{2}$ for correctly counting $\emptyset\emptyset$s and $11$s in part (b).
  (The substring $\emptyset\emptyset$ appears $\max\{0, 2n - 1\}$ times in the string $\emptyset^{2n}$, not $n$ times!)

• $\frac{1}{2}$ for the divisibility argument in part (c)
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.

**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 \( \text{palin}(L) \) as follows:

\[
\begin{align*}
Q' &:= (Q \times Q) \cup \{ s' \} \\
s' &\text{ is an explicit state in } Q' \\
A' &:= \{ (q, q) \mid q \in Q \} \\
\delta'(s', \epsilon) &:= \{ (s, q) \mid q \in A \} \\
\delta'((p, q), a) &:= \{ (\delta(p, a), q') \mid \delta(q', a) = q \}
\end{align*}
\]

\( M' \) reads its input string \( w \) and simulates \( M \) reading the input string \( ww^R \). Specifically, \( M' \) simulates two copies of \( M \), one running forward from the start state \( s \), and the other running backward starting from an accept state.

- The new start state \( s' \) non-deterministically guesses the final accept state of \( M \) on input \( ww^R \).
- State \((p, q)\) means that the forward (left) copy of \( M \) is in state \( p \), and the backward (right) copy of \( M \) is in state \( q \).
- \( M' \) accepts if and only if the forward simulation of \( M \) on \( w \) and the backward simulation of \( M \) on \( w^R \) meet at the same halfway state.

Rubric: 5 points =
+ 1 for a formal, complete, and unambiguous description of a DFA or NFA
  - No points for the rest of the problem if this is missing.
+ 3 for a correct NFA
  - –1 for a single mistake in the description (for example a typo)
+ 1 for a brief English justification. We explicitly do not want a formal proof of correctness, but we do want one or two sentences explaining how the NFA works.
(b) Prove that the language \( drome(L) := \{ w \mid w^Rw \in L \} \) 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 \( drome(L) \) as follows:

\[
\begin{align*}
Q' & := (Q \times Q) \cup \{s'\} \\
s' & \text{ is an explicit state in } Q' \\
A' & = \{(s, q) \mid q \in A\} \\
\delta'(s', \epsilon) & = \{(q, q) \mid q \in Q\} \\
\delta'((p, q), a) & = \{(p', \delta(q, a)) \mid \delta(p', a) = p\}
\end{align*}
\]

\( M' \) reads its input string \( w \) and simulates \( M \) reading the input string \( w^Rw \). Specifically, \( M' \) non-deterministically guesses a halfway state \( h \), and then simulates two copies of \( M \), one running forward starting at \( h \), and the other running backward starting at \( h \).

- The new start state \( s' \) non-deterministically guesses the halfway state \( h \).
- State \((p, q)\) means that the backward (left) copy of \( M \) is in state \( p \), and the forward (right) copy of \( M \) is in state \( q \).
- \( M' \) accepts if and only if the backward simulation of \( M \) on \( w^R \) ends at the start state \( s \), and the forward simulation of \( M \) on \( w \) ends at an accepting state.

**Rubric:** 5 points =

- 1 for a formal, complete, and unambiguous description of a DFA or NFA
  - No points for the rest of the problem if this is missing.
- 3 for a correct NFA
  - −1 for a single mistake in the description (for example a typo)
- 1 for a brief English justification. We explicitly do not want a formal proof of correctness, but we do want one or two sentences explaining how the NFA works.
Solved problem

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

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

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

\[
\square
\]

Rubric: 5 points =

+ 1 for a formal, complete, and unambiguous description of a DFA or NFA
  - No points for the rest of the problem if this is missing.
+ 3 for a correct NFA
  - −1 for a single mistake in the description (for example a typo)
+ 1 for a brief English justification. We explicitly do not want a formal proof of correctness, but we do want one or two sentences explaining how the NFA works.