1. **Building NFAs.** For each of the languages below, construct an NFA that accepts the language. You may either draw the NFA or write out a formal transition function. In either case, you need to label/explain your states and briefly argue why the NFA accepts the correct language.

(a) All strings over $\Sigma = \{0, 1\}$ that have two of the same characters at a distance 3 from each other. E.g., 101011, 10100.

**Solution:**

\[
\begin{array}{c}
\text{start} \\
q_0 \\
q_1 & \xrightarrow{0,1} & q_2 & \xrightarrow{0,1} & q_3 \\
q_5 & \xrightarrow{0,1} & q_6 & \xrightarrow{0,1} & q_7 \\
t & \xrightarrow{0,1} & t
\end{array}
\]

$q_0$: Start state; we have not read the first matching character
$q_1$: We read first matching character and it's a 1
$q_2$: We read any symbol so we have 1x.
$q_3$: We read any symbol so we have 1xx.
$q_4$: We read first matching character and it's a 0
$q_5$: We read any symbol so we have 0x.
$q_6$: We read any symbol so we have 0xx.
t: We have found the matching characters at distance 3, any remaining suffix is ok

(b) All strings over $\Sigma = \{0, 1, \ldots, 9\}$ that contain both 374 and 473 as substrings.

**Solution:**

\[
\begin{array}{c}
\text{start} \\
q_0 & \xleftarrow{\Sigma} & q_1 \xrightarrow{3} q_2 \xleftarrow{\Sigma} q_3 \xrightarrow{4} q_4 \xleftarrow{\Sigma} q_5 \xrightarrow{3} q_0 \\
q_6 & \xleftarrow{\Sigma} & q_7 \xrightarrow{3} q_8 \xleftarrow{\Sigma} q_9 \xrightarrow{7} q_10 \xrightarrow{4} t
\end{array}
\]
2. **NFAs to DFAs.** For the following regular expressions, do the following steps:
- Construct an NFA corresponding to the regular expression using Thompson’s algorithm
- Use the incremental subset construction to convert the NFA to a DFA
- Create another DFA with fewer states to recognize the language

(a) \((\varepsilon + 0)(1 + 10)^*\)

i. NFA:

![NFA Diagram]

ii. DFA from incremental subset construction:
iii. DFA:
This language basically means there cannot be two consecutive 0s.

\[
\begin{array}{c|c|c|c|c}
q' & \epsilon - \text{reach}(q') & q' \in A? & \delta'(q', 0) & \delta'(q', 1) \\
\hline
a & abcd f ghikp & \checkmark & e & jl \\
e & ef ghikp & \checkmark & \emptyset & jl \\
jl & hjlmop & \checkmark & n & jl \\
n & hiknop & \checkmark & \emptyset & jl \\
\end{array}
\]

Note: the above table was not required but helps understand the solution.

\[
\begin{array}{c|c|c|c}
\text{a} & 00 & \text{b} & 00 \text{ seen and last char a 0} \\
\text{b} & 00 & \text{c} & 00 \text{ has been seen in the input} \\
\end{array}
\]

(b) \(0^*(10^*0^*)^*\)

i. NFA:
ii. DFA from incremental subset construction:

<table>
<thead>
<tr>
<th>$q'$</th>
<th>$\varepsilon - \text{reach}(q')$</th>
<th>$q' \in A$?</th>
<th>$\delta'(q',0)$</th>
<th>$\delta'(q',1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$abcdef$</td>
<td>✓</td>
<td>$c$</td>
<td>$g$</td>
</tr>
<tr>
<td>$c$</td>
<td>$abcdef$</td>
<td>✓</td>
<td>$c$</td>
<td>$g$</td>
</tr>
<tr>
<td>$g$</td>
<td>$ghikl$</td>
<td>✓</td>
<td>$j$</td>
<td>$m$</td>
</tr>
<tr>
<td>$j$</td>
<td>$hi j k l$</td>
<td>✓</td>
<td>$j$</td>
<td>$m$</td>
</tr>
<tr>
<td>$m$</td>
<td>$ef noqr$</td>
<td>✓</td>
<td>$p$</td>
<td>$g$</td>
</tr>
<tr>
<td>$p$</td>
<td>$ef nopqr$</td>
<td>✓</td>
<td>$p$</td>
<td>$g$</td>
</tr>
</tbody>
</table>

iii. DFA:

The strings in the language should have even number of 1s.
3. **Palindromes.** In both subproblems below, you need to formally specify the NFA $N$ and formally prove that it accepts the language required by the problem.

(a) Given a DFA $M$, define an NFA $N$ such that $L(N) = \{x \in L(M) | x = x^R\}$, i.e., $N'$ accepts the strings in $L(M)$ that are palindromes.

**Solution:** This is impossible; e.g., if $L(M) = \Sigma^*$, then $L(N)$ is the language of all palindromes, which is not regular and therefore cannot be matched by an NFA. ■

(b) Given a DFA $M$, define an NFA $N$ such that $L(N) = \{x \in \Sigma^* | xx^R \in L(M)\}$

**Solution:** Let $M = (\Sigma, Q, \delta, s, A)$ be the given DFA. We construct the NFA $N = (\Sigma, Q', \delta', s', A')$ from $M$ as follows.

$$Q' = (Q \times Q) \cup \{s'\}$$

$s'$ is an explicit state in $Q'$

$$A' = \{(h, h) | h \in Q\}$$

$$\delta'(s', \epsilon) = \{(s, a) | a \in A\}$$

$$\delta'((p, q), a) = \{(\delta(p, a), q') | \delta(q', a) = q\} \quad \text{for } p, q \in Q, a \in \Sigma$$

$N$ simultaneously simulates two copies of $M$ on the input string: one that runs normally and one that runs in reverse. To run the normal copy of $M$ on some input symbol, $N$ simply chooses the next state as defined by $M$’s transition function. To run the reverse copy of $M$ on some input symbol, $N$ non-deterministically guesses the previous state from which taking the input symbol transition leads to the current state in the reverse copy.

- The new start state $s'$ non-deterministically guesses the accepting state $a = \delta^*(s, ww^R)$ without reading any input.
- State $(p, q)$ means the following:
  - The current state resulting from executing $M$ on the input string $w$ starting from state $s$ is now $p$.
  - The guess for the current state resulting from executing $M$ in reverse on the input string $w$ starting at some accepting state $a \in A$ is now state $q$.
- $N$ accepts $w$ if and only if the input string $w$ leads both the normal and reverse copy of $M$ to some "halfway" state $h \in Q$.

We can formally prove that $N$ accepts the correct language.
Lemma 1. For any \( n \geq 0 \), if \( x \in \Sigma^* \) with \( |x| = n \), then for any \( q_1, q_2 \in Q \):
\[
\delta^*(q_1, q_2, x) = \{(q_3, q_4) \in Q \times Q | \delta^*(q_1, x) = q_3, \delta^*(q_4, x^R) = q_2\}
\]

Proof: Suppose the lemma holds for all \( w \in \Sigma^* \) with \( |w| < |x| \). If \( |x| = 0 \) then \( x = \epsilon \). \( \delta^*((q_1, q_2), \epsilon) = \{(q_1, q_2)\} \) (note that the only \( \epsilon \) transition is from \( s' \)). Since \( \delta^*(q_1, \epsilon) = q_1 \) and \( \delta^*(q_2, \epsilon^R) = \delta^*(q_2, \epsilon) = q_2 \), the lemma holds for this case.

If \( |x| > 0 \) then \( x = aw \) for \( a \in \Sigma, w \in \Sigma^* \) with \( |w| < |x| \). Then
\[
\delta^*((q_1, q_2), x) = \delta^*(w, a) = \bigcup_{(q', q'') \in \delta^*((q_1, q_2), a)} \delta^*((q', q''), w) = \bigcup_{q'' \in Q, \delta(q'', a) = q_2} \delta^*((q_1, q''), w)
\]
By the inductive hypothesis, we know that
\[
\delta^*(\delta(q_1, a), q'') = \{(q_3, q_4) \in Q \times Q | \delta^*(\delta(q_1, a), w) = q_3, \delta^*(q_4, w^R) = q''\}
\]
But \( \delta^*(\delta(q_1, a), w) = \delta^*(q_1, aw) = \delta^*(q_1, x) \). Likewise, since \( \delta(q'', a) = q_2 \), then
\[
\delta^*(q_4, x^R) = \delta^*(q_4, w^R a) = \delta(\delta(q_4, w^R), a) = \delta(q'', a) = q_2
\]
This proves the lemma.

Now suppose that for some \( x \in \Sigma^* \), \( xx^R \in L(M) \). Then \( \delta^*(s, x) = h \) for some \( h \in Q \) and \( \delta^*(h, x^R) = a \) for some \( a \in A \). By the lemma, \( (h, h) \in \delta^*(s, x) \). Since \( (s, a) \in \epsilon - \text{reach}(s') \), we have \( (h, h) \in \delta^*(s', x) \). And since \( (h, h) \in A' \), we have \( x \in L(N) \).

Conversely, suppose that \( x \in L(N) \). Then for some \( h \in Q \), \( (h, h) \in \delta^*(s', x) \). If \( x = \epsilon \), then
\[
\delta^*(s', x) = \{s'\} \cup \{(s, a) | a \in A\}
\]
Therefore \( (h, h) = (s, a) \) for some \( a \in A \), which implies that \( s \in A \) and so \( xx^R = \epsilon = e \in L(M) \).

If \( x \neq \epsilon \) then \( x = cw \) for some \( c \in \Sigma \) and \( w \in \Sigma^* \). Then:
\[
\delta^*(s', x) = \delta^*(s', cw) = \bigcup_{p \in \epsilon - \text{reach}(s')} \bigcup_{r \in \delta'(p, c) \delta^*(r, w)}
\]
Since \( s' \) only has \( \epsilon \) transitions, there must be some state \( (s, a) \in \epsilon - \text{reach}(s') \) such that:
\[
(h, h) \in \bigcup_r \delta((s, a), c) \delta^*(r, w) = \delta^*((s, a), cw) = \delta^*((s, a), x)
\]
By the lemma, \( \delta^*(s, x) = h \) and \( \delta^*(h, x^R) = a \), so \( \delta^*(s, xx^R) = a \). Since \( a \in A \), we have \( xx^R \in L(M) \).

4. Not to submit: Recall that for any language \( L \), \( \overline{L} = \Sigma^* - L \) is the complement of \( L \). In particular, for any NFA \( N \), \( \overline{L(N)} \) is the complement of \( L(N) \).

Let \( N = (Q, \Sigma, \delta, s, A) \) be an NFA, and define the NFA \( N_{\text{comp}} = (Q, \Sigma, \delta, s, Q \setminus A) \). In other words we simply complemented the accepting states of \( N \) to obtain \( N_{\text{comp}} \). Note that if \( M \) is DFA then \( M_{\text{comp}} \) accepts \( \Sigma^* - L(M) \). However things are trickier with NFAs.
(a) Describe a concrete example of a machine \( N \) to show that \( L(N_{\text{comp}}) \neq L(N) \). You need to explain for your machine \( N \) what \( L(N) \) and \( L(N_{\text{comp}}) \) are.

(b) Define an NFA that accepts \( L(N) - L(N_{\text{comp}}) \), and explain how it works.

(c) Define an NFA that accepts \( L(N_{\text{comp}}) - L(N) \), and explain how it works.

*Hint:* For all three parts it is useful to classify strings in \( \Sigma^* \) based on whether \( N \) takes them to accepting and non-accepting states from \( s \).
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:

- $Q' = (Q \times Q \times Q) \cup \{s'\}$
- $s'$ 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 $ww$. 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.

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.