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., 101101, 10100.

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

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) $(\epsilon + 0)(1 + 10)^*$

   (b) $0^*(10^*10^*)^*$

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

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

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 $L(N_{\text{comp}}) \neq \overline{L(N)}$. You need to explain for your machine $N$ what $\overline{L(N)}$ and $L(N_{\text{comp}})$ are.

   (b) Define an NFA that accepts $\overline{L(N)} - L(N_{\text{comp}})$, 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 $s$. 

---

**CS/ECE 374 ♦ Fall 2019**  
**Homework 2 ♦**  
Due Tuesday, September 17, 2019 at 8 p.m.
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.

\[\square\]

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.