Problem 1 Let $L = \{w \mid w \text{ contains at least two } 0 \text{ and at most one } 1\}$. Construct an NFA that recognizes $L^*$. 

Problem 2 Prove that every NFA can be converted to an equivalent one that has a single accept state.

Problem 3 Prove that regular languages are closed under complement. [Hint: given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes a language $L$, build a new DFA $M' = (Q', \Sigma, \delta', q'_0, F')$ that recognizes $L' = \Sigma^* \setminus L = \{w \mid w \not\in L\}$.

Problem 4

a. Use the result from Problem 3 along with other closure properties of regular languages to show that regular languages are closed under set difference. That is, given regular languages $L_1$ and $L_2$, show that

$$L_1 \setminus L_2 = \{w \in L_1 \mid w \not\in L_2\}$$

is regular.

b. Show that regular languages are closed under symmetric set difference

$$L_1 \triangle L_2 = \{w \mid \text{either } w \in L_1 \text{ or } w \in L_2 \text{ but not both}\}.$$ 

Problem 5

a. For any language $L$, we defined

$$\text{PREFIX}(L) = \{w \mid \exists x \in \Sigma^* \text{ s.t. } wx \in L\}.$$ 

Prove that regular languages are closed under PREFIX.

b. For any language $L$, we defined

$$\text{SUFFIX}(L) = \{w \mid \exists x \in \Sigma^* \text{ s.t. } xw \in L\}.$$ 

Using closure properties of regular languages and the result of part a, prove that regular languages are closed under SUFFIX.
Problem 6

a. Let $\Sigma$ and $\Gamma$ be alphabets and let $f : \Sigma \to \Gamma$ be a function that maps symbols in $\Sigma$ to symbols in $\Gamma$. One such example is $f : \{1, 2, 3, 4, 5\} \to \{a, b, c, d\}$ given by

\[
\begin{align*}
  f(1) &= b \\
  f(2) &= b \\
  f(3) &= a \\
  f(4) &= d \\
  f(5) &= a
\end{align*}
\]

We can extend such an $f$ to operate on strings $w = w_1w_2 \cdots w_n$ by $f(w) = f(w_1)f(w_2) \cdots f(w_n)$. Using the same example, $f(132254) = \text{babbad}$. We can extend $f$ to operate on languages by $f(L) = \{f(w) \mid w \in L\}$.

Prove that if $L$ is a regular language and $f : \Sigma \to \Gamma$ is an arbitrary function—that is, it is not necessarily the example given above—then $f(L)$ is regular.

[Hint: given a DFA $M$ that recognizes $L$, build an NFA $N$ that recognizes $f(L)$ by applying $f$ to the symbols on each transition.]

b. A homomorphism is a function $f : \Sigma \to \Gamma^*$ that maps symbols in $\Sigma$ to strings over $\Gamma$. One example of a homomorphism is the function that maps every string to $\varepsilon$. A less-trivial example is $f : \{a, b\} \to \{a, b, c\}$ given by

\[
\begin{align*}
  f(a) &= \text{bacca} \\
  f(b) &= b.
\end{align*}
\]

As before, we can extend $f$ to operate on strings $w = w_1w_2 \cdots w_n$ by $f(w) = f(w_1)f(w_2) \cdots f(w_n)$ and languages by $f(L) = \{f(w) \mid w \in L\}$.

Prove that regular languages are closed under homomorphism. [Hint: As with your construction in part a, you want to apply $f$ to the symbols on each transition but in this case you may need to add additional states if the length of $f(a)$ is not 1. Be sure to handle the case where $f(a) = \varepsilon$.]

Problem 7 For languages $L_1$ and $L_2$, define

\[ L_1 \ominus L_2 = \{w \in L_1 \mid w \text{ does not contain any string in } L_2 \text{ as a substring}\}. \]

Prove that regular languages are closed under $\ominus$. \[\text{[Hint: Think about what } \Sigma^* \circ L \circ \Sigma^* \text{ means for a language } L. \text{ Write } L_1 \ominus L_2 \text{ in terms of set difference and concatenation and apply closure properties of regular languages.]}\]

Problem 8 For each language in Exercise 1.6 in Sipser, give an equivalent regular expression. (You don’t need to prove that it’s correct.)

---

1You can typeset $\ominus$ in \LaTeX{} by putting the line `\usepackage{mathabx}` in the preamble and using `\backslash` in math mode.
**Problem 9** Using the procedure given in Lemma 1.55 in Sipser, convert the regular expression $(0 \cup 11)^*01(00 \cup 1)^*$ to an NFA. Show each step.

**Problem 10** Using the procedure given in Lemma 1.60 in Sipser, convert the following DFA to a regular expression. Show each step.