Problem 1: Define the following words, phrases and symbols.

1. Finite state machine, finite automaton
   Set of states, rules for transitioning between them, alphabet, start state and accepting states.

2. Sipser page 49. Determinism vs Nondeterminism
   Determinism: There is one way for the computation to proceed at each step/on each input
   Non-determinism: There can be several choices for how to proceed at each step/on each input

3. Sipser page 54. DFA vs NFA
   DFA - Single transition per character from each state
   NFA - Multiple transitions per character allowed, ε transitions allowed

4. Regular Language
   A language recognized by a finite automaton

5. Sipser page 53. \((Q, \Sigma, \delta, q_0, F)\)
   \(Q\) is a finite set of states
   \(\Sigma\) is a finite alphabet
   \(\delta: Q \times \Sigma \rightarrow P(Q)\) is the transition function
   \(q_0 \in Q\) is the start state
   \(F \subseteq Q\) is the set of accept states

6. \(\emptyset\)
   The empty language, the language containing no strings

7. \(\varepsilon\)
   The empty string

8. Epsilon Transition
   A transition made without consuming any input. In the case of an NFA it can be considered as if the
   machine splits into multiple copies with one machine staying in the current state other following each
   of the ε labeled arrows.

   Union, Concatentation, "*".

10. Sipser page 36,40. A machines can accept many strings, but only a single language. To avoid confusion,
    we will usually say a machine accepts a string and recognizes a language.

Problem 2: Are the following statements true or false?

1. It is possible for a finite automaton to recognize an infinite language. True

2. Every deterministic finite automaton is also a nondeterministic finite automaton. True

3. For every nondeterministic finite automaton, there is an equivalent nondeterministic finite automaton
    that has no epsilon transitions. True. See Sipser p.56.
4. For every nondeterministic finite automaton, there is an equivalent nondeterministic finite automaton that has a single accept state. True

5. If you swap the accept states and the reject states on ANY finite automaton, the new machine will recognize the complement of the original language. False. Consider the following two NFAs which accept the same language:

\[
\begin{align*}
Q &= \{q_0, q_1\}, \Sigma = \{0, 1\}, \delta(q_0, \varepsilon) = q_1, \delta(q_1, 0) = q_0, \delta(q_1, 1) = q_0, q_0 = q_0, F = \{q_1\} \\
Q &= \{q_0, q_1\}, \Sigma = \{0, 1\}, \delta(q_0, \varepsilon) = q_1, \delta(q_1, 0) = q_0, \delta(q_1, 1) = q_0, q_0 = q_0, F = \{q_0\}
\end{align*}
\]

6. The class of languages recognized by non-deterministic finite automata is closed under complementation. True

7. The class of languages recognized by non-deterministic finite automata is not closed under set difference. False. Rewrite set difference as a combination of complement and intersect

8. Sipser problem 1.24. For any string \( w = w_1w_2 \ldots w_n \), the reverse of \( w \), written \( w^R \), is the string \( w \) in reverse order, \( w_n \ldots w_2w_1 \). For any language \( A \) let \( A^R = \{w^R | w \in A\} \). If \( A \) is regular, then so is \( A^R \).

   True. Make new start state with epsilon transitions to all accept states. Make start state the only accept state.

Problem 3: Create finite automata for each of the following languages over the alphabet \( \Sigma = \{0, 1\} \). Give a characterization for each state.

1. The language of strings that do not contain an odd number of 1s.
2. The language of strings that contain '0110' and end in '1'.
3. The language of strings that contain a '0' and don’t end in '10'.
4. The language of part 2 concatenated with that of part 3.

Problem 4: Let’s prove that the following automaton recognizes exactly the language \( L = \{w \in \{0, 1\}^* | w \text{ contains a 10 and end in 1}\} \). To do this, we will need to prove that our FA (1) accepts all strings in \( L \) and (2) does not accept any string not in \( L \).

1. Characterize each state.
2. Forward direction (accepts all strings in \( L \)). Proof by Induction?
3. Reverse direction (does not accept any string outside of \( L \)). Proof by Contradiction?

Solution 4: The machine recognizes the language \( L = \{x | x \text{ contains a 10 and ends in 1}\} \).
1. Suppose $x$ is in $L$. Then $x$ is of the form $y_1$ for some $y$, such that $y$ has 10 as a substring (Intuitively, here we disconnect the two conditions that lead to acceptance. Note that it might not always be possible to do this).

Now, we prove that $\delta^*(1, y) \in \{3, 4\}$. This suffices to prove our claim, since from any of the states in $\{3, 4\}$, if the machine reads a 1, it goes to the accept state (and therefore, the string $x = y_1$ is accepted).

Consider the first occurrence of the substring 10 in $y$. Write $y$ as $y_110y_2$, where $y_1$ does not contain any 10. (Intuitively, $y_1$ is either the string of all 0s or a string formed by some 0s followed by some 1s. We prove by induction on the length of $y_1$ that if $y_1$ does not contain a 10, then $\delta^*(1, y_1) \in \{1, 2\}$. This is certainly true for an empty string. The induction hypothesis is that,

- If a string $z$ ends in a 0, $\delta^*(1, z) = 1$
- If a string $z$ ends in a 1, $\delta^*(1, z) = 2$.

Assume the induction hypothesis for a string $z$ length $k$.

- Assume $y_1$ ends in 0. Therefore, $y_1 = z0$ for some $z$ of length $k$. Then, since $y_1$ does not contain a 10, $z$ must end in a 0. Therefore, $\delta^*(1, y_1) = \delta(\delta^*(1, z), 0) = \delta(1, 0) = 1$.
- Now, assume $y_1 = z1$. Now, $z$ could end in a 0 or a 1. Say $z$ ended in a 0. Then $\delta^*(1, y_1) = \delta(\delta^*(1, z), 1) = \delta(1, 1) = 2$ (Because $\delta^*(1, z) = 1$ if $z$ ends in 0). Say $z$ ended in a 1. Then $\delta^*(1, y_1) = \delta(\delta^*(1, z), 1) = \delta(2, 1) = 2$. Thus, the induction hypothesis is true for $y_1$ too.

Now, note that from any of the states $\{1, 2\}$, after reading a 10, the machine lands up in state 3. Therefore, $\delta^*(1, y_110) = 3$. After reaching one of the states in $\{3, 4\}$, the machine cannot go back to any of the states in $\{1, 2\}$. Therefore, $\delta^*(1, y_110y_2) \in \{3, 4\}$. Furthermore, $\delta^*(1, x) = \delta^*(1, y_110y_21) = \delta(\{3, 4\}, 1) = 4$. (Notational Clarification: By $\delta(S, a)$ for a set $S \subseteq Q$, we mean $\bigcup_{q \in S} \delta(q, a)$).

2. Now suppose $M$ accepts a string $x$ not in $L$. Either $x$ does not end in a 1 or it does not contain a substring 10. Clearly, if $x$ does not end in a 1 it does not reach the accept state, state 4. On the other hand, suppose it does not contain 10 as a substring. (Go back to the previous proof and reuse parts of it). How? Remember, we showed that if a string $z$ does not contain a 10, it ends in one of the states $\{1, 2\}$. We are done. ■

**Problem 5:** Show that the language $L_5 = \{s \in \{0, 1\}^* \mid s \text{ divisible by 5}\}$ is regular.

**Problem 6:** Optional (if we have enough time)
(Closure of regular languages under perfect shuffle) (Sipser 1.41)
For languages $A$ and $B$, let the **perfect shuffle** of $A$ and $B$ be the language:

$$\{w \mid w = a_1b_1a_2b_2\ldots a_kb_k, \text{ where } a_1a_2\ldots a_k \in A, b_1b_2\ldots b_k \in B.\}$$

Show that regular languages are closed under the perfect shuffle operation.

**Problem 7:** Optional (if we have enough time)
An all-paths-NFA $M$ is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ that accepts $x \in \Sigma^*$ if and only if every possible state that $M$ could be in after reading $x$ is a state from $F$. Prove that all-NFAs recognize exactly the regular languages. (Notice the contrast with NFAs)

**Solution 7:** Note, first, that an all-paths-NFA has the same syntax as an NFA, but their acceptance criteria are different. An NFA accepts a string $x$ if and only if at least one of the computation paths leads to an accepting state. On the other hand, an all-paths-NFA accepts a string if and only if every computation path ends in an accepting state. In particular, the same machine can act as an NFA and an all-paths-NFA, depending on the acceptance criterion it uses.

Now, on to the solution: $M$ is an all-paths-NFA. The gameplan is to show that, for every $L$ that is accepted by some all-paths-NFA $M$, there is an NFA $N$ that accepts $L$. This will prove that, if $L$ is accepted by some
all-paths-NFA, \( \bar{L} \) is regular. Since the class of regular languages is closed under complement, \( L \) is regular too.

Let \( L = L(M) \). We show that the NFA \( N = (Q, \Sigma, \delta, q_0, F' = Q - F) \), accepts \( \bar{L} \). Suppose \( x \in L \). Then, by definition, all the computation paths of \( M \) on \( x \) leads to a state in \( F \). Which means none of the computation paths end in \( F' = Q - F \). Therefore, \( N \) (an NFA) does not accept \( x \), and \( x \not\in L(N) \). It is not hard to see that, if \( x \not\in L \) (that is, it is not accepted by \( M \)), then \( x \) is accepted by \( N \), and therefore \( x \in L(N) \). (We leave this as an exercise). We have shown that, \( x \in L(M) \) if and only if \( x \not\in L(N) \). Therefore, \( N \) accepts \( \bar{L} \). \( \blacksquare \)