Problem 2.

(a) The given language \( L_1 \) is in the form \( Q \in \{0, 1, 11, 113\} \).

(b) Similar problems were studied in class.

Step 1: Draw an NFA where \( L_1 \times \) is the longest suffix of \( x \).

Step 2: On consuming \( T \), \( \epsilon \) is a prefix of the pattern string 111.

Consider the language \( L_2 = \{ z \in \{0, 1\}^* \mid x \text{ ends with } 113 \} \).

The acceptance of \( L_2 \) by a deterministic finite automaton was studied in class.

For example, input states to remember the two ending symbols \( x \).

Assume such DFA \( M_2 \) accepts \( L_2 \).

Then, we obtain \( M \) by complementing the set of final states.

Assume such DFA \( M_2 \) accepts \( L_2 \).

Then, we obtain \( M \) by complementing the set of final states.
Problem 3.
(a) Basic idea: The machine nondeterministically guesses (when reading an input symbol of) \(0\) or \(1\) as a sub-string of \(0(0+1)^2+0\) that is forthcoming.

\[ M: \quad \text{start} \rightarrow \text{\(q_{\text{start}}\)} \rightarrow 0 \rightarrow \text{\(q_1\)} \rightarrow \text{\(q_{\text{odd}}\)} \rightarrow \text{\(q_{\text{even}}\)} \rightarrow \text{\(q_{\text{11}}\)} \]

\(q_{\text{start}}\): nondeterministically wait or guess on an input symbol \(0, 1\)

\(q_1, q_{\text{odd}}, q_{\text{even}}, q_{\text{11}}\): having encountered an input symbol \(0, 1\), verify if a sub-string of no form \(0(0+1)^2+0\) appears - can be regular expressions!

Can verify that \(\forall x \in \{0, 1\}^*\), \(M\) accepts \(x\) iff \(x \in (0+1)^* (0(0+1)^2) (0(0+1)^2)^+ \).

(b) The given language is the disjoint union of the two languages:
\[
L_a = \{ x \in \{a,b,c\}^* \mid \#_a(x) > 3 \text{ and } 0 \leq \#_b(x), \#_c(x) \leq 2 \}
\]
\[
L_b = \{ x \in \{a,b,c\}^* \mid \#_a(x) > 3 \text{ and } 0 \leq \#_a(x), \#_c(x) \leq 2 \}
\]

Basic idea for constructing a DFA \(M_a\) accepting \(L_a\): each state has 3 components to record \(\#_a(x), \#_b(x), \text{ and } \#_c(x)\) in the input consumed so far.

\(Q = \{ (i,j,k) \in \mathbb{N}^3 \mid i \leq 3, j \leq 2, 0 \leq k \leq 3 \} \cup \{q\} \)

\(N = \{0, 1, 2, \ldots\}\)

Start state: \((0, 0, 0)\)

Set of accepting states: \(\{(3, j, k) \mid 0 \leq j, k \leq 2\}\)
1-step transition function \( s : Q \times \{a, b, c\} \rightarrow Q \) is defined as:

\[
\begin{align*}
 s((i, j, k), a) &= ((i+1, j, k), \text{if } i \leq 2) \\
 s((i, j, k), b) &= ((i, j+1, k), \text{if } j \leq 1) \\
 s((i, j, k), c) &= ((i, j, k+1), \text{if } k \leq 1) \\
 &\quad \quad \quad \quad \quad \quad \text{if } \text{exceed}
\end{align*}
\]

\[ \forall \text{ state } q \in \{a, b, c\}, \quad s(q, \text{exceed}) = \text{exceed} \]

A DFA \( M_b \) accepting \( L_b \) is similar.

A desired FA accepting \( L_a \cup L_b \) is:

\[
\text{start} \rightarrow \circ \xrightarrow{a} M_a \xrightarrow{\varepsilon} \circ \xrightarrow{b} M_b
\]

(c) Given that an FA \( M \) accepting \( L \) (without loss of generality, we may assume that \( M \) has one accepting state \( q_{\text{accept}} \)), we construct an FA \( M' \) accepting \( \text{halfl}(L) \).

The problem was studied in class with an NFA \( M' \) that guesses the ending states of \( M \) for the input \( x \) to \( M' \) at the beginning, then uses two “forward simulations” to verify the guess, in one (to arrive at guess state) or continue subsequent guesses in the other (to arrive an accepting state \( q_{\text{M'}} \)).

Here, we use a different idea: \( M' \) keeps track of 2 states in \( M \) using two coordinates (tracks in \( q \) state in \( M' \)).
For each input symbol read in $M'$, $M'$ uses first coordinate/fallback to simulate $M$ on that symbol. (At the same time, $M'$ simulate the backward simulation starting at $q_{accept}$ in $M$.) Simultaneously, $M'$ uses second coordinate/fallback to simulate $M$ backwards on a guessed symbol.

$M'$ accepts an input $x$ if the forward simulation (on $x$) and the backward simulation (on a guessed $y$, $|y| = |x|$) are in a common state of $M$.

Formally, assume that NFA $M = (Q, \Sigma, \delta, q_0, q_{accept})$ accepts $L$.

Construct an NFA $M' = (Q', \Sigma, \delta', q_0', q_{accept}')$ as follows: $Q' = Q \times Q$, $q_0' = (q_0, q_{accept})$, $F' = \{ (q, q) \mid q \in Q \}$, and

$$\delta' : Q' \times \Sigma \to 2^Q$$

is defined as:

$$\delta'( (p, q), a) = \{ (r, s) \in Q' \mid r \in \delta(p, a) \text{ and } \exists b \in \Sigma \exists q' \in \delta(s, b) \}$$
Problem 5. (Similar to an exercise in class and in Homework 1).

\[ L = \{ 0^n 1^n \mid n \geq 0 \} \]

We show that there does not exist any DFA accepting \( L \).

Suppose the contrary that \( L = L(M) \) for some DFA \( M = (Q, \Sigma, \delta, q_0, F) \), where \( Q = \{ q_1, q_2, \ldots, q_n \} \) for some positive integer \( n \).

Consider the sequence of strings:

- \( x_1 = 0^1 \)
- \( x_2 = 0^2 \)
- \( \vdots \)
- \( x_n = 0^n \)
- \( x_{n+1} = 0^{n+1} \)

By Pumping Lemma, there exist \( i, j \in \{1, 2, \ldots, n+1\} \) such that \( i \neq j \) at the two inputs \( 0^i \) and \( 0^j \) cause two identical versions of \( M \), starting from \( q_1 \), to be in the same state, say \( p \in Q \).

By the diagram:

- Start \( \rightarrow q_1 \)
- Data labeled by \( 0^i \)
- Data labeled by \( 0^j \)

Now, consider suffixing \( i \) to augment the two input strings \( 0^i \) and \( 0^j \) to \( 0^i 1^i \) and \( 0^j 1^j \), respectively.

And notice that the augmentation \( 1^i \) causes the two versions of \( M \) to a common state \( p'' \) as well. (Why?)
The input string $0^i10^j$ is a palindrome ($\in \mathcal{L}$), so $M$ should accept $0^i10^j$, i.e., $p' \in F$. But, the input string $0^i10^j (i \neq j)$ is not a palindrome ($\notin \mathcal{L}$), so $M$ should reject $0^i10^j$, i.e., $p' \notin F$, a contradiction!
Problem 6

The basic idea of constructing a DFA $N$ is that it essentially mimics the behavior of $M$, but in addition, $N$ keeps track of a bit that indicates if the state $q$ has been visited.

The bit starts out as 0, and is flipped to 1 in the event that $r$ is reached. The bit is never flipped back once it turns to 1.

The accepting states of $N$ are of the form $(1, q)$ where $q \in F$ as they indicate that $M$ is in an accepting state $(q, F)$ and the state $r$ has been visited.

$$N = \left( \{0, 1\} \times Q, \Sigma, s', (0, q_0), \{(1, q) | q \in F\} \right)$$

where

$$s' : (\{0, 1\} \times Q) \times \Sigma \rightarrow (\{0, 1\} \times Q)$$

is defined as:

$$s' ((0, q), a) = \begin{cases} (0, s(q, a)) & \text{if } s(q, a) \neq r \\ (1, s(q, a)) & \text{if } s(q, a) = r \end{cases}$$

and

$$s' ((1, q), a) = (1, s(q, a))$$
Problem 7.
(Props were given in class.)

Assume that \( K = \mathcal{L}(M) \) for some DFA \( M = (Q, \Sigma, \delta, q_0, F) \).
Define an NFA \( M' = (Q \cup \{103\}, \Sigma, \eta, 103, F) \) where
the transition function \( \eta \) is defined as:
\[
\eta(q_0, \varepsilon) = \{ q \in Q \mid \text{there exists } w \in \Sigma^* \text{ such that } w \text{ drives } M \\
\text{from } q_0 \text{ to } q \text{ via } \delta \}
\]
(set of all reachable states from the start state \( q_0 \) by \\
setting \( q_0 \) to \( q \) by following \( \varepsilon \)-or more \ntransitions specified by \( \delta \)
and \( \eta(q, \alpha) = \bigcup \delta(q, \alpha) \) for every \( q \in Q \)
and \( \alpha \in \Sigma \),
and \( \eta \) takes the value \( \emptyset \) in all other cases.

Can you see that \( \mathcal{L}(M') = K \) ?

For \( \#2 \): Define a DFA \( M_2 = (Q, \Sigma, \delta, q_0, P) \):
\( M_2 \) is the same as \( M \) except that its accepting 
states (in \( P \)) are all \( q \) in \( M \) from which
it is possible to reach an accepting state
in \( M \):
\[
P = \{ q \in Q \mid \text{there exists } w \in \Sigma^* \text{ and } r \in F \\
\text{such that } w \text{ drives } M \text{ from } q \text{ to } r \text{ via } \delta \}
\]
Problem 8: The statement is false.

Consider the following NFA with 8 states; it accepts every string $x \in \Sigma^*$ (say $\Sigma = \{0,1\}$) can modify our construction for $\Sigma = \{0,1,3\}$ with $|x| \leq 11$, but does not accept $0^{12}$. 