Problem 2.

(a) Basic idea: The machine nondeterministically guesses (when reading an input symbol 0) or waits for a substring of $0(0+1)^*0$ that is forthcoming. $0,1$.

$M$: Start $\xrightarrow{o} a \xrightarrow{0,1} 0 \xrightarrow{0,1} 0 \xrightarrow{0,1} 0 \xrightarrow{0} 0$.

$q_{start}$: nondeterministically wait or guess on an input symbol $0$.

$q_1$, $q_2$, $q_{false}$, $q_{true}$: having encountered an input symbol $0$, verify if a substring of the form $0(0+1)^*0$ appears.

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

(b) The given language is the disjoint union of the two languages:

$L_a = \{ x \in \{ 0, 1, c \}^* \mid \#(x) > 3 \text{ and } 0 \leq \#(x), \#(c(x)) \leq 2 \}$

$L_b = \{ x \in \{ 0, 1, c \}^* \mid \#(x) > 2 \text{ and } 0 \leq \#(x), \#(c(x)) \leq 2 \}$.

Basic idea for constructing a DFA $M_a$ accepting $L_a$: each state has 3 components to record $\#(y) \#(x), \#(z)$ in the input consumed so far.

$Q = \{ (i, j, k) \in N^3 \mid i \leq 3, j \leq 2, \text{ and } \#(y) \leq 3 \}$.

$N = \{0, 1, 2, \ldots \}$

Start state: $(0, 0, 0)$

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

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

For \( \delta(a,b,c) \), \( \sigma (\text{exceed, exceed}) = \text{exceed} \).

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

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

\[
\begin{align*}
\text{start} &\rightarrow Q_a &\rightarrow M_b
\end{align*}
\]

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

The basic idea is that \( M' \) keeps track of two states in \( M \) (using two coordinates/tracks in a state of \( M' \)): 
The "forward simulation" is defined as follows:

For each input symbol read in \( M' \), \( M' \) uses its first and second coordinate/track to simulate \( M \) on that symbol.

(At the same time, \( M' \) simulates the backward simulation starting at \( q_{accept} \).)

The "backward simulation" is as follows:

Simultaneously, \( M' \) uses its second coordinate/track 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 \( M' = (Q', \Sigma, \delta', q'_0, F') \) as follows:

\( Q' = Q \times \overline{Q} \), \( q'_0 = (q_0, q_{accept}) \),

\( F' = \{(p, q) \mid q \in Q \overline{3} \} \),

and \( \delta' : Q' \times \Sigma \to 2^{Q'} \) is defined as:

\[ \delta'(p, q) \circ a = \delta(Q', q_0, \overline{Q} \overline{S}) \circ a \]

forward simulation

\[ \delta'(p, q, a) = \delta(Q', r, \overline{Q} \overline{S}) \circ a \]

guessed symbol

\[ \delta'(p, q, a) = \delta(Q', q, \overline{Q} \overline{S}(s, b)) \]

backward simulation
1.38 Use the same construction given in the proof of Theorem 1.39, which shows the equivalence of NFAs and DFAs. We need only change $F'$, the set of accept states of the new DFA. Here we let $F' = \mathcal{P}(F)$. The change means that the new DFA accepts only when all of the possible states of the all-NFA are accepting.

1.21 In both parts we first add a new start state and a new accept state. Several solutions are possible, depending on the order states are removed.

a. Here we remove state 1 then state 2 and we obtain
\[a^*b(a \cup ba^*b)^*\]

b. $\Sigma^*0\Sigma^* \cup 1111\Sigma^* \cup 1 \cup \varepsilon$

l. $(1\Sigma)^*(1 \cup \varepsilon)$

Annotate the regular expression.

1.29 b. Let $A_2 = \{www | w \in \{0,1\}^*\}$. We show that $A_2$ is nonregular using the pumping lemma. Assume to the contrary that $A_2$ is regular. Let $p$ be the pumping length given by the pumping lemma. Let $s$ be the string $a^pba^pb$. Because $s$ is a member of $A_2$ and $s$ has length more than $p$, the pumping lemma guarantees that $s$ can be split into three pieces, $s = xyz$, satisfying the three conditions of the lemma. However, condition 3 implies that $y$ must consist only of $a$s, so $xyyz \notin A_2$ and one of the first two conditions is violated. Therefore $A_2$ is nonregular.

1.53 Assume to the contrary that $ADD$ is regular. Let $p$ be the pumping length given by the pumping lemma. Choose $s$ to be the string $1^p0+1^p$, which is a member of $ADD$. Because $s$ has length greater than $p$, the pumping lemma guarantees that $s$ can be split into three pieces, $s = xyz$, satisfying the conditions of the lemma. By the third condition in the pumping lemma have that $|xy| \leq p$, it follows that $y$ is $1^k$ for some $k \geq 1$. Then $xy^2z$ is the string $1^{p+k-1}0+1^p$, which is not a member of $ADD$, violating the pumping lemma. Hence $ADD$ isn't regular.
(a) \[ L = \{ uu^Rv \mid u, v \in (01)^* \} \] is regular, since 
\[ L \] is denoted by a regular expression 
\[ 0(01)^*0 \cup 1(01)^*1. \]

To see that \( L \subseteq L(0(01)^*0 \cup 1(01)^*1) \):

Let \( x \in L \) be arbitrary, i.e., \( x = uu^Rv \) for some \( u, v \in (01)^* \).

Since \( u \in (01)^* \), \( u = 0u' \) or \( u = 1u' \) for some \( u' \in (01)^* \).

Assume \( u = 0u' \) (the case for \( u = 1u' \) is similar).

Then \( x = uu^Rv = 0u'v \in (01)^*1 = 0(01)^*1 \subseteq 0(01)^*1 \).

To see that \( L(0(01)^*0 \cup 1(01)^*1) \subseteq L \):

Let \( x \in L(0(01)^*0 \cup 1(01)^*1) \) be arbitrary.

Assume \( x \in 0(01)^*1 \) (the case for \( x \in 1(01)^*1 \) is similar).

Then \( x = uu^Rv \) where \( u = 0 \) and \( v \in (01)^*1 \).

That is, \( x \in L \).

(b) \[ L = \{ uu^Rv \mid u, v \in (01)^* \} \] is not regular. Suppose that it were.

We can apply the Pumping Lemma directly on \( L \). Here we use closure properties for regularity first to "reduce" \( L \) into \( L' \):

Consider \( L' = L \cap 1(0^2)^*1 (1^2)^*011 \).

Certainly \( L' \) would be regular since "\( \cap \)" preserves regularity.

But, what is \( L' \) (or, why do we consider \( \cap 1(0^2)^*1 (1^2)^*011 \))?

Try \( x \in L' \).

The string \( x \) is of the form \( uu^Rv \):

Hence \( L' = \{ 10^{2t+1} 1110^{2t+1} \mid n > 0 \} \)

Now, apply the Pumping Lemma on \( L' \) (remember, there are many cases to check).
Problem 9.

(a) We prove the non-regularity of \( L = \{ w \in \{a, b\}^* \mid \#_a(w) = \#_b(w) \} \) by contradiction via the closure properties (a regular)

Suppose that \( L \) were regular.

Then \( L = \{a, b\}^* - L \) would be regular.

Regular closure

\( (a+b)^* \)

preserving

- What is \( L \)?

\[ L = \{ w \in \{a, b\}^* \mid \#_a(w) = \#_b(w) \} \]

Hence, we consider

\[ L \cap \{a, b\}^* = \{ w \in \{a, b\}^* \mid \#_a(w) = \#_b(w) \} \]

Regular closure

\( (a+b)^* \)

preserving

\[ = \{a^i b^i \mid i \geq 0 \} \]

a non-regular

Thus, \( L \) is not regular.