Problem 2.

(a) Basic idea: The machine nondeterministically guesses (when reading an input symbol 0) at what a substring of $0(0+1)^+0$ that is forthcoming $0, 1$

$$M: \quad \text{Start} \rightarrow Q_{\text{start}} \quad 0 \rightarrow Q_1 \quad 0, 1 \quad \text{Good} \rightarrow Q_{\text{even}} \quad 0 \rightarrow Q_{\text{end}}$$

$Q_{\text{start}}$: nondeterministically wait or guess on an input symbol $0$.

$Q_1$, $Q_{\text{even}}$, $Q_{\text{end}}$: having encountered an input symbol $0$, verify if a substring of the form $0(0+1)^+0$ appears.

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

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

$L_a = \{ x \in (a, b, c)^* | \#(a(x)) \geq 3 \text{ and } 0 \leq \#(b(x)), \#(c(x)) \leq 2 \}$

$L_b = \{ x \in (a, b, c)^* | \#(a(x)) \geq 3 \text{ and } 0 \leq \#(b(x)), \#(c(x)) \leq 2 \}$

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

$Q = \{ (i, j, k) \in \mathbb{N}^3 | i \leq 3, j \leq 2, 0 \leq k \leq 3 \}$

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:

$$s((i, j, k), a) = \begin{cases} (i+1, j, k) & \text{if } i \leq 2 \\ (i, j, k) & \text{if } i = 3 \end{cases}$$

$$s((i, j, k), b) = \begin{cases} (i, j+1, k) & \text{if } j \leq 1 \\ \text{fail} & \text{if } j = 2 \end{cases}$$

$$s((i, j, k), c) = \begin{cases} (i, j, k+1) & \text{if } k \leq 1 \\ \text{fail} & \text{if } k = 2 \end{cases}$$

For all $i, j, k \in \{a, b, c\}$, $s(\text{fail}, d) = \text{fail}$.

A DFA $M_b$ accepting $L_b$ is similar.

A desired FA accepting $L_a \cup L_b$ is:

```
start \rightarrow (q_0) \xrightarrow{\varepsilon} M_a \xrightarrow{\varepsilon} 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 half ($L$).

The basic idea is that $M'$ keeps track of two states in $M$ (using two coordinates/track in a state of $M'$).
For each input symbol read in $M'$, $M'$ uses first coordinate/make to simulate $M$ on that symbol.

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

Simultaneously, $M'$ uses second coordinate/make to simulate $M$ backwards on a guessed symbol.

$M'$ accepts on input $x$ iff 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: $Q' = Q \times Q$, $q'_0 = (q_0, q_{accept})$, $F' = \{(q, q) | q \in Q \}$, and $\delta' : Q' \times \Sigma \to Q'$ is defined as:

$\delta'(p, q) \in Q' \iff q_0 \in \delta(p, a)$ and $\exists b \in \Sigma, q \in \delta(s, b)$. 

$\delta'(p, q), a = \{ (r, s) \in Q \times Q | r \in \delta(p, a) \text{ and } \exists b \in \Sigma, q \in \delta(s, b) \}$. 

Forward Simulation

Backward Simulation

Guessed Symbol
Problem 3. (Similar to Homework 1, problem 9)

Let \( L = \{ x \in \{0,1\}^* \mid x^r = x^3 \} \)

We show that there do 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

\[
\begin{align*}
x_1 &= 0^n \\
x_2 &= 0^{n+1} \\
&\vdots \\
x_n &= 0^{n+n} \\
x_{n+1} &= 0^{n+n+1}
\end{align*}
\]

By Pigeonhole Principle, there exist \( i, j \in \{1, 2, \ldots, n+1\} \) such that \( i \neq j \) and 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 \( p \in Q \).

\[
\text{start} \xrightarrow{0^i} q_1 \xrightarrow{0^n} q \xrightarrow{0^{n+n}} q \xrightarrow{0^{n+n+1}} p
\]

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

- The augmentation \( 1 \cdot 0^i \) causes the two versions of \( M \\
 0^i 1 \cdot 0^j \) to a common state \( p' \), as well. (Why?)

But \( \ldots \)
The input string $0^i 1^j 0^i$ is a palindrome ($\in L$), so $M$ should accept $0^i 1^j 0^i$, i.e., $p' \in F$. But, the input string $0^i 1^j 0^i$ (i.e., $i+j$) is not a palindrome ($\notin L$), so $M$ should reject $0^i 1^j 0^i$, i.e., $p' \notin F$, a contradiction!
Problem 6

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 $r$ has been visited.

The bit starts out as 0, one 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.

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

where

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

defined as:

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

and

$$\delta'((1, q), a) = (1, \delta(q, a))$$
Problem 5
Try it again.
Careful computation of ε-closures.

Problem 6
Same as textbook Problem 1.38

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.