1. (Problem 2.22)

(a) The proof is by contradiction, so assume that the language \( L = \{a^nba^{2n} \mid n \in \mathbb{N}\} \) is accepted by some FA \( M \). Let \( n \) be the number of states in \( M \).

Now consider the string \( x = a^nba^{2n} \). The pumping lemma tells us that \( x = uvw \) for some \( u, v, w \) satisfying: (i) \( |uv| \leq n \), (ii) \( v \neq \Lambda \), and (iii) \( uv^iw \in L \) for all \( i \in \mathbb{N} \).

Notice (because the first \( n \) letters of \( x \) are all \( a \)'s) that \( v \) must be equal to \( a^k \) for some \( k \). Furthermore, condition (ii) tells us that \( k > 0 \).

Next, I will contradict condition (iii) by showing that \( uv^0w \not\in L \). To see this, notice that \( uv^0w = a^{n-k}ba^{2n} \) (because \( v = a^k \)). The number of \( a \)'s on the left side is \( n - k \), and the number of \( a \)'s on the right side is \( 2n \). But \( 2(n - k) \neq 2n \) (because \( k > 0 \)), so we conclude that \( uv^0w \not\in L \).

As explained above, this produces a contradiction; therefore, our assumption (that \( L \) was accepted by \( M \)) must have been false.

(b) As usual, the proof is by contradiction: assume that there exists a machine \( M \) accepting \( L = \{a^i b^j a^k \mid k > i + j\} \). Let \( n \) be the number of states in \( M \).

Consider the string \( x = a^nba^{n+2} \). (Clearly, \( x \in L \) because \( n + 2 > n + 1 \).) The pumping lemma tells us that \( x = uvw \) for some \( u, v, w \) satisfying the usual conditions.

As usual, we see that \( v = a^m \) for some \( m > 0 \) (thanks to conditions (i) and (ii)).
Now, to contradict condition (iii), I will show that \( uv^2w \notin L \). To see this, notice that \( uv^2w = a^{n+m}ba^{n+2} \). The only possible way to match this string to the \( a^i b^j a^k \) pattern is to let \( i = n + m \), \( j = 1 \), and \( k = n + 2 \). But then \( m > 0 \) implies \( n + 2 \leq n + m + 1 \), so \( k \leq i + j \). This shows that \( uv^2w \notin L \), as desired.

2. The proof is by contradiction, so assume that the language \( L = \{a^n ba^{2n} \mid n \in \mathbb{N}\} \) is accepted by some FA. The language is clearly infinite (each \( n \in \mathbb{N} \) is uniquely associated with a member of \( L \)), so we can use the weaker pumping lemma to say that there exist strings \( u, v, w \) satisfying (i) \( |v| > 0 \), and (ii) \( uv^i w \in L \) for all \( i \in \mathbb{N} \).

In particular, \( uv^0w = uw \) must be in the language. So \( uw = a^m ba^{2m} \) for some \( m \).

Now, to contradict condition (ii), I will show that \( uvw \notin L \). There are two cases to consider:

- Case where \( v \) contains at least one \( b \): We have already seen that \( uw \) contains exactly one \( b \), so we conclude that \( uvw \) must contain at least two \( bs \). But every member of \( L \) has exactly one \( b \), so \( uvw \notin L \).

- Case where \( v \) consists only of \( a \): In this case, \( v = a^k \) for some \( k \in \mathbb{N} \). Furthermore, condition (i) guarantees \( k > 0 \).

Now, we have already seen that \( uw = a^m ba^{2m} \). Therefore, by considering all possible values of \( u \) and \( w \), we see that either \( uvw = a^{m+k} ba^{2m} \) or \( uvw = a^m ba^{2m+k} \). But either way, \( uvw \) cannot be in \( L \) because \( k > 0 \) implies \( 2(m + k) \neq 2m \) and \( 2m \neq 2m + k \).

3. (Problem 2.27)

(a) Use the product construction to create a machine for \( L(M_1) \cup L(M_2) \). Next, use depth-first search to look for a path from the start state to a non-accepting state. If you find one, then the answer is yes; otherwise, the answer is no.

This works because if such a path exists, then the letters on the path will form a string accepted by neither \( M_1 \) nor \( M_2 \).

(c) As shorthand, let us use \( L \) to refer to the language accepted by the machine \( M \).
First, use the minimization algorithm on $M$ to produce a new machine $M^*$. Next, run the string $x$ through the machine $M^*$, and then run $y$ through $M^*$. If they end up in different states, then the answer is yes; otherwise, the answer is no.

This works because $I_{M^*} = I_L$—in other words, two strings will end up in the same state of the minimal machine if and only if they are indistinguishable with respect to the machine’s language.

(d) First, run the string $x$ through the machine $M$; let us use $q$ to refer to the state reached by $x$. Next, use depth-first search to look for a path from $q$ to an accepting state. If you find one, then the answer is yes; otherwise, the answer is no.

This works because if such a path exists, then the letters on the path will form a string $z$ such that $xz \in L$.

(g) Use the product construction to create a machine $M_3$ for $L(M_1) - L(M_2)$. If $L(M_3)$ is empty, then the answer is yes; otherwise, the answer is no.

To decide whether $L(M_3)$ is empty, simply use depth-first search on $M_3$ to look for a path from the start state to an accepting state. If you fail to find one, then $L(M_3)$ is empty; otherwise, $L(M_3)$ is not empty.

This works because $L(M_1) - L(M_2) = \emptyset$ if and only if $L(M_1) \subseteq L(M_2)$.

4. (Problem 2.29)

(a) False. Let $L_1 = \{a^n b^n \mid n \in \mathbb{N}\}$ and let $L_2 = \{a, b\}^*$. Clearly, $L_1 \subseteq L_2$ because $L_2$ contains all strings. Additionally, we know that $L_1$ is not accepted by any FA, but we can easily create a one-state FA for $L_2$.

(b) False. Let $L_1 = \emptyset$ and let $L_2 = \{a^n b^n \mid n \in \mathbb{N}\}$. Clearly, $L_1 \subseteq L_2$ because $L_1$ is empty. Additionally, we know that $L_2$ is not accepted by any FA, but we can easily create a one-state FA for $L_1$.

(d) False. Let $L_1 = \{a^n b^n \mid n \in \mathbb{N}\}$ and let $L_2 = \{b^n a^n \mid n \in \mathbb{N}\}$. It is not hard to see that these languages are not accepted by FA. However, $L_1 \cap L_2 = \{\Lambda\}$ is accepted by a two-state FA.
(e) True. The proof is by contraposition, so assume that $L'$ is accepted by some FA. I will prove that $L$ is accepted by some FA.

Since $L'$ is accepted by some FA, we can use the closure properties to see that $L''$ is also accepted by some FA. But $L'' = L$, so we conclude that $L$ is accepted by some FA.

(i) False. Let $L_n = \{a^n b^n\}$ for every $n \in \mathbb{N}$. Clearly, $|L_n| = 1$ for every $n$, and finite languages are accepted by FA. However, $\bigcup_{n=1}^{\infty} L_n = \{a^n b^n \mid n \in \mathbb{N}\}$, which we know is not accepted by any FA.

5. (Problem 2.43) For every string $x$, it is the case that $\{x\}$ is an $I_L$ equivalence class.

It suffices to prove that if $x \neq y$ then $x$ is $L$-distinguishable from $y$. Let us assume without loss of generality that $|x| \leq |y|$. Now there are three cases:

Case 1. Assume $|x| = |y|$. In this case, we can let $z = x$. Then $xz = xx \in L$; it remains to prove that $yz = yx \notin L$.

Since $|x| = |y|$, we know that $y$ and $x$ are the two halves of the string $yx$. But we already know that $x \neq y$; therefore, $yx \notin L$ because its left half differs from its right half.

Case 2. Assume $|x| < |y|$ and the parity of $|x|$ differs from the parity of $|y|$. In this case, again we can let $z = x$. Then $xz = xx \in L$, but $yz = yx \notin L$ because $|yx|$ is odd. This prevents $yx$ from being of the form $ww$.

Case 3. Assume $|x| < |y|$ and the parity of $|x|$ is the same as the parity of $|y|$. (As shorthand, let us use $m$ and $n$ to refer to $|x|$ and $|y|$ respectively.) In this case, we can let $z = ab^n xab^n$. Obviously, $xz = xab^n xab^n \in L$; it remains to show that $yz \notin L$.

To prove this, we must find the middle of the string $yz$ so that we can split it in half and compare the two halves. To do this, let $k = \frac{n+m}{2}$ (this is definitely an integer because $m$ and $n$ have the same parity). Then we can split $yz$ into two halves:

- $yab^k$, whose length is $n + 1 + k = n + 1 + \frac{n+m}{2}$, and
- $b^{n-k} xab^n$, whose length is $n - k + m + 1 + m = n + 1 + \frac{n+m}{2}$. 


To see that the two halves differ, look at the \((k + 1)\)st letter from the end. In the first string, it is obviously an \(a\). However, the corresponding letter in the second string is a \(b\) because \(k < n\) (because \(m < n\)).

This establishes the fact that \(yz \notin L\), as desired.

6. (Problem 2.55)

(e) After running the minimization algorithm, we get the following table:

\[
\begin{array}{c|cccccccc}
2 & \times \\
3 & \times & \times \\
4 & \times & \times & \times \\
5 & \times & \times & \times & \times \\
6 & \times & \times & \times & \times & \times \\
7 & \times & \times & \times & \times & \times & \times & \times \\
\hline
1 & 2 & 3 & 4 & 5 & 6
\end{array}
\]

Every cell is marked, which means that no two states can be safely merged together. Therefore, the original machine is already minimal.

(f) After running the minimization algorithm, we get this table:

\[
\begin{array}{c|cccccccc}
2 & \times \\
3 & \times \\
4 & \times & \times & \times \\
5 & \times & \times \\
6 & \times & \times & \times & \times \\
7 & \times & \times & \times & \times & \times & \times \\
\hline
1 & 2 & 3 & 4 & 5 & 6
\end{array}
\]

According to the table, the states 1, 3, 5 and 7 can be merged, as well as the states 2 and 6. Then the minimal machine is

![Minimal Machine Diagram]
7. (Extra credit) (Problem 2.51)

Solution 1.

Let $L_1$ be a language accepted by some FA, and let $L_2$ be some other language.

To prove that $L_1/L_2$ is accepted by FA, I will show that the number of $I_{L_1/L_2}$ equivalence classes is no more than the number of $I_{L_1}$ equivalence classes. Then the finiteness of the latter will guarantee the finiteness of the former, which in turn will guarantee the existence of some FA for $L_1/L_2$.

To prove that the number of $I_{L_1/L_2}$ equivalence classes is no more than the number of $I_{L_1}$ equivalence classes, I will show that distinguishability with respect to $L_1/L_2$ implies distinguishability with respect to $L_1$.

To this end, let us assume $x$ and $y$ are strings that are distinguishable with respect to $L_1/L_2$. This means that for some string $z$, we have (without loss of generality) $xz \in L_1/L_2$ while $yz \notin L_1/L_2$.

By the definition of $L_1/L_2$, this means that there exists a string $w \in L_2$ such that $xzw \in L_1$; similarly, we know that $yzw \notin L_1$ because $yz \notin L_1/L_2$.

Now, to distinguish $x$ and $y$ with respect to $L_1$, we can extend them with the string $zw$. Thanks to the previous paragraph, this shows us that $x$ and $y$ are $L_1$-distinguishable, as desired.

Solution 2.

Let $L_1$ be a language accepted by some FA $M$, and let $L_2$ be some other language. I will construct a machine $M'$ accepting $L_1/L_2$.

$M'$ is defined to be the same as $M$, except that the final states are different. To explain the difference, let $A$ and $A'$ be the sets of final states in machines $M$ and $M'$ respectively; then $A'$ is defined by

$$A' = \{ q \in Q \mid \exists y \in L_2, \delta^*(q, y) \in A \}.$$

It remains to prove that $M'$ accepts $L_1/L_2$. To do so, I will show that
an arbitrary string \( x \) is in \( L_1/L_2 \) if and only if \( M' \) accepts \( x \):

\[
x \in L_1/L_2 \\
\iff \exists y \in L_2, \ xy \in L_1 \quad \text{def'n of } L_1/L_2 \\
\iff \exists y \in L_2, \ \delta^*(q_0, xy) \in A \quad \text{def'n of machine acceptance} \\
\iff \exists y \in L_2, \ \delta^*(\delta^*(q_0, x), y) \in A \quad \text{identity on p.54} \\
\iff \delta^*(q_0, x) \in A' \quad \text{def'n of } A' \\
\iff M' \text{ accepts } x \quad \text{def'n of machine acceptance.}
\]