1. (a) $ab\Sigma^*ba$
   (b) $\lambda + a + b + (aa + ba + bb)\Sigma^*$

2. $aab$ is the shortest string which is not in the language denoted by the regular expression $b^*(abb^*)^*a^*$.

3. (a) NFA for the marble toy (Color coded for edge intersections):
We can describe this language \( L \) with an NFA \( M = (Q, \Sigma, \delta, LLL, F) \) such that \( L = L(M) \) where \( Q = \{LLL, LLR, LRL, LRR, RLL, RLR, RRL, RRR, LLL', LRL', RLR', RLL', RRL'\} \), \( \Sigma = \{0, 1\} \), \( F = \{LLR, RRL, LLL', LRL', RLR', RLL', RRL'\} \) and \( \delta \) as:

\[
\begin{array}{|c|c|c|c|}
\hline
\delta & 0 & 1 & \lambda \\
\hline
LLL & RLL & LRR & \emptyset \\
LLR & RLR & LLL' & \emptyset \\
LRL & RRL & LRR & \emptyset \\
LRR & RRR & LRL & \emptyset \\
RLL & LRL & RRR & \emptyset \\
RLR & LRR & RLL & \emptyset \\
RRL & LLL' & RLR' & \emptyset \\
RRR & LLR & RRL & \emptyset \\
LLL' & \emptyset & \emptyset & LLL \\
LRL' & \emptyset & \emptyset & LRL \\
RLR' & \emptyset & \emptyset & RLR \\
RLL' & \emptyset & \emptyset & RLL \\
RRL' & \emptyset & \emptyset & RRL \\
\hline
\end{array}
\]
(b) We can describe the language $L$ by some patterns of 0 and 1’s that come before the last input. Because $x_3$ is only affected by inputs from B, we can see that each word that ends with 1 and contains an even number of 1’s is in the language. If $x_2$ is pointing right by 2 or 3 inputs from A, and $x_3$ is pointing left as default, the next 1 marble will come out of D. If $x_2$ is not modified by inputs from A, $x_2$ and $x_3$ can be set to right and left respectively by 2 inputs from B.

We will describe number of inputs from B as “# of 1’s in u mod 4” because every 4 inputs from A or B set the related levers into its original position, so it doesn’t affect the system.

We can describe this as a language like this:

$L_1 = \{u \mid (# \text{ of } 1’s \text{ in } u \text{ is odd})$
\[\lor ((# \text{ of } 0’s \text{ in } u \text{ mod } 4) \geq 2 \land (# \text{ of } 1’s \text{ in } u \text{ mod } 4) = 0) \lor ((# \text{ of } 0’s \text{ in } u \text{ mod } 4) \leq 1 \land (# \text{ of } 1’s \text{ in } u \text{ mod } 4) = 2)\}$

For words that end with 0, $x_1$ and $x_2$ both need to point right in the previous state for the word to be accepted. If there’s one marble passed from A (for $x_1$) and 1 or 2 marbles have passed from B, the $x_1$ and $x_2$ get in the aforementioned state. If three marbles have passed from A, both $x_1$ and $x_2$ are in correct position for the marble from A. If $x_2$ is in right position, if there are 1 or 2 marbles from B, it switches to left position, so there needs to be 0 or 3 marbles from B as previous inputs for $x_2$ to be in correct position.

$L_0 = \{u0 \mid ((# \text{ of } 0’s \text{ in } u \text{ mod } 4) = 1$
\[\land 1 \leq (# \text{ of } 1’s \text{ in } u \text{ mod } 4) \leq 2) \lor ((# \text{ of } 0’s \text{ in } u \text{ mod } 4) = 3$
\[\land ((# \text{ of } 1’s \text{ in } u \text{ mod } 4) = 3 \lor (# \text{ of } 1’s \text{ in } u \text{ mod } 4) = 0))\}$

Finally, we can describe L as union of two previous languages.

$L = L_0 \cup L_1$

4. $0^*1^*$

5. (a) DFA of $f^{-1}(L)$:

(b) $(abb)^*baa + abb(baa)^*abb$

(c) $(0 + 1)0^*(1(0 + 1)0^*)^*$
6. (a) It’s false. We can give \( L_1 = \{ a^n b^n \mid n \geq 0 \} \) and \( L_2 = \Sigma^* \) as counterexamples. Here \( L_1 \subseteq L_2 \) and \( L_1 \) is not regular. But \( L_2 \) is regular.

(b) It’s false. We can give \( L_1 = \{ab\} \) and \( L_2 = \{a^n b^n \mid n \geq 0\} \) as counterexamples. Here \( L_1 \subseteq L_2 \) and \( L_2 \) is not regular. But \( L_1 \) is regular.

(c) It’s false. We can give \( L_1 = \{a^n b^n \mid n \geq 0\} \) and \( L_2 = \Sigma^* - L_1 \) as counterexamples. Here \( L_1 \) and \( L_2 \) are not regular. But \( L_1 \cup L_2 = \Sigma^* \) is regular.

(d) It’s false. We can give \( L_1 = \{a^n b^n \mid n \geq 0\} \) and \( L_2 = \{b^n a^n \mid n \geq 0\} \) as counterexamples. Here \( L_1 \) and \( L_2 \) are not regular. But \( L_1 \cap L_2 = \{\lambda\} \) is regular.

(e) It’s false. We can give \( L_1 = \{ab\} \) and \( L_2 = \{a^n b^n \mid n \geq 0\} \) as counterexamples. Here \( L_1 \) is regular and \( L_2 \) is not regular. But \( L_1 \cap L_2 = \{ab\} \) is regular.

(f) It’s false. We can give \( L_1 = \{ab\}, L_2 = \{a^2 b^2\}, \ldots, L_n = \{a^n b^n\} \) as counterexamples. Here \( L_1, L_2, \ldots, L_n \) are all regular. But \( \cup_{n=1}^{\infty} L_n = \{a^n b^n \mid n > 0\} \) is nonregular.

(g) It’s true. If \( L_1 \) is regular and \( L_1 \) is finite, then \( L_1 L_2 \) is regular. Both \( L_1 \) and \( L_2 \) is a regular language and regular languages are closed under concatenation, therefore \( L_1 L_2 \) is regular.

(h) It’s true. If \( L_1 \cup L_2 \) is regular and \( L_1 \) is finite, then \( L_2 \) is regular. We can write \( (L_1 \cup L_2) - L_1 = L_2 - L_1 \) and \( L_2 = (L_1 \cap L_2) \cup (L_2 - L_1) \). Both \( L_1 \) and \( L_1 \cup L_2 \) are regular, therefore \( L_2 - L_1 \) is regular because set difference is closed under regular languages. \( L_1 \cap L_2 \) is regular because since \( L_1 \) is finite, the intersection is finite, therefore regular. \( L_2 \) is union of two regular languages, therefore it is regular.

(i) It’s false. We can give \( L_1 = \{\lambda, a\}, L_2 = \{a^n \mid n \text{ is a composite number}\} \), and \( L_1 L_2 = \{a^n \mid n \geq 4\} \) as counterexamples. In this case, \( L_1 L_2 \) is regular, \( L_1 \) is finite but \( L_2 \) is nonregular.
7. We can find the right quotient of $L$ with $L_2 = \Sigma$ (all letters of the alphabet). $L/L_2$ will contain all strings $x$ where $xy$ is in $L$ and $y$ is in $L_2$. Because $L_2$ contains all letters in alphabet, $x$ will be all words with right-most letter missing from it. This is the definition of $\text{truncate}$ operator, therefore $L/L_2 = \text{truncate}(L)$. Theorem 4.4 in the book states right quotient is closed under regular languages. We know $L_2$ is regular because it is finite. So for any regular language $L$, $\text{truncate}(L)$ is regular because of closure property of right quotient.

8. Our algorithm takes a word $w$ and a regular language $L$ such that $w \in L$ and length of $w$ is even and determines whether $w^R \in L$. If $w$ and $w^R$ are in $L$, then $w$ and $w^R$ are in $L^R$ as well. We know $L$ has a corresponding DFA. From that DFA, we can construct an NFA corresponding to $L^R$ by reversing the edge directions, making the old start state the new final state and creating a new starting state and creating edges from that starting state to old final states by a $\lambda$-transition. If this NFA of $L^R$ accepts $w$, this means $w \in L^R$ which implies $w^R \in L$. This algorithm works for any length of $w$.

9. We’ll use pumping lemma to show $L = \{a^mb^k \mid m > k\}$ is nonregular. For any selection of $m$, we can choose $w = a^{m+1}b^m = xyz$ where $|xy| \leq m$ and $|y| \geq 1$ and every $w_i = xy^iz$ for all $i \in \mathbb{N}$ is in $L$. In this case, $xy = a^m$ and $y = a^k$ where $1 \leq k \leq m$. $w_0 = a^{m+1-k}b^m$ and $m+1-k \leq m$ for all $k$, therefore $w_0$ is not in $L$ for any $y$ and $L$ is not regular.

10. We’ll use pumping lemma to show $L = \{a^m! \mid m \geq 0\}$ is nonregular. For any selection of $m$, we can choose $w = a^m! = xyz$ where $|xy| \leq m$ and $|y| \geq 1$ and every $w_i = xy^iz$ for all $i \in \mathbb{N}$ is in $L$. In this case, $xy = a^m$ and $y = a^k$ where $1 \leq k \leq m$. $w_0 = a^{m!-k}$ and $m! > m! - k > (m - 1)!$, therefore $w_0$ is not in $L$ for any $y$ and $L$ is not regular.