1. Give a regular expression for each of the languages below:

(a) \( L = \{ w \mid w \text{ contains an even number of 0s} \}. \Sigma = \{0, 1\} \)

(b) \( L = \{ w \mid w \text{ is any string except } a \text{ and } b \}. \Sigma = \{a, \ldots, z\} \)

(c) \( L = \{ w \mid w \text{ is any string that doesn’t contain exactly two } a \text{’s} \}. \Sigma = \{a, \ldots, z\} \)

(d) \( L = \{ w \mid w \text{ is a palindrome of length less than four} \}. \Sigma = \{a, b\} \)

2. Problem 1.20 (Regexp membership)

3. Problem 1.7 b,e,h (Designing NFAs)

4. Problem 1.16 (NFA→DFA conversion)

5. Problem 1.28 (regexp→NFA conversion)

6. Problem 1.21 (NFA→regexp conversion)

(For this problem, you can use the T1/T2/T3 method given in class; you don’t have to use the similar, but slightly different, procedure described in the book.)

7. The following language \( L \) is composed of the intersection of two simpler languages \( L_1 \) and \( L_2 \). Construct a NFA for each of the two simpler languages and perform the NFA intersection needed to create the NFA for the full language \( L \). The alphabet for the languages is over \( \{0, 1\} \).

\( L_1 = \{ w \mid w \text{ either starts with a 0 or ends with a 0 } \} \).

\( L_2 = \{ w \mid w \text{ contains an odd number of 0s } \} \).

\( L = L_1 \cap L_2 \)
8. Now, do NFA intersection in the general case. Suppose we have two NFAs, 
NFA_1 = (Q_1, \Sigma, \delta_1, s_{0,1}, F_1) and NFA_2 = (Q_2, \Sigma, \delta_2, s_{0,2}, F_2). Show how to 
define a “cross-product” NFA whose language is L(NFA_1) \cap L(NFA_2).

Don’t solve this problem by converting the machines to DFAs and working 
there. Define the components of the new NFA (its Q, \delta, etc.) directly in 
terms of the components of NFA_1 and NFA_2.

Hint: Work out a solution without worrying about \varepsilon-transitions, then figure 
out how to extend your solution to handle these.