Problem 1: What language \( L \) does the following automaton recognize? Prove that the automaton indeed recognizes the language you think it recognizes. To do this, we will need to prove that our FA (1) accepts all strings in \( L \) and (2) does not accept any string not in \( L \).

1. Forward direction (accepts all strings in \( L \)).
2. Reverse direction (does not accept any string outside of \( L \)).

Solution 1: The machine recognizes the language \( L = \{ x \mid x \text{ contains a 10 and ends in 1} \} \).

1. Suppose \( x \) is in \( L \). Then \( x \) is of the form \( y1 \) for some \( y \), such that \( y \) has 10 as a substring (Intuitively, here we disconnect the two conditions that lead to acceptance. Note that it might not always be possible to do this).

Now, we prove that \( \delta^*(1, y1) \in \{3, 4\} \). This suffices to prove our claim, since from any of the states in \( \{3, 4\} \), if the machine reads a 1, it goes to the accept state (and therefore, the string \( x = y1 \) is accepted). Consider the first occurrence of the substring 10 in \( y \). Write \( y \) as \( y1 \ 10y2 \), where \( y1 \) does not contain any 10. (Intuitively, \( y1 \) is either the string of all 0s or a string formed by some 0s followed by some 1s). We prove by induction on the length of \( y1 \) that if \( y1 \) does not contain a 10, then \( \delta^*(1, y1) \in \{1, 2\} \). This is certainly true for an empty string. The induction hypothesis is that,

- If a string \( z \) ends in a 0, \( \delta^*(1, z) = 1 \)
- If a string \( z \) ends in a 1, \( \delta^*(1, z) = 2 \).

Assume the induction hypothesis for a string \( z \) length \( k \).

- Assume \( y1 \) ends in 0. Therefore, \( y1 = z0 \) for some \( z \) of length \( k \). Then, since \( y1 \) does not contain a 10, \( z \) must end in a 0. Therefore, \( \delta^*(1, y1) = \delta(\delta^*(1, z), 0) = \delta(1, 0) = 1 \).

- Now, assume \( y1 = z1 \). Now, \( z \) could end in a 0 or a 1. Say \( z \) ended in a 0. Then \( \delta^*(1, y1) = \delta(\delta^*(1, z), 1) = \delta(1, 1) = 2 \) (Because \( \delta^*(1, z) = 1 \) if \( z \) ends in 0). Say \( z \) ended in a 1. Then \( \delta^*(1, y1) = \delta(\delta^*(1, z), 1) = \delta(2, 1) = 2 \). Thus, the induction hypothesis is true for \( y1 \) too.

Now, note that from any of the states \( \{1, 2\} \), after reading a 10, the machine lands up in state 3. Therefore, \( \delta^*(1, y1 10) = 3 \). After reaching one of the states in \( \{3, 4\} \), the machine cannot go back to any of the states in \( \{1, 2\} \). Therefore, \( \delta^*(1, y1 10y2) \in \{3, 4\} \). Furthermore, \( \delta^*(1, x) = \delta^*(1, y1 10y2) = \delta(\{3, 4\}, 1) = 4 \). (Notational Clarification: By \( \delta(S, a) \) for a set \( S \subseteq Q \), we mean \( \bigcup_{q \in S} \delta(q, a) \).)

2. Now suppose \( M \) accepts a string \( x \) not in \( L \). Either \( x \) does not end in a 1 or it does not contain a substring 10. Clearly, if \( x \) does not end in a 1 it does not reach the accept state, state 4. On the other hand, suppose it does not contain 10 as a substring. (Go back to the previous proof and reuse parts of it. How?) Remember, we showed that if a string \( z \) does not contain a 10, it ends in one of the states \( \{1, 2\} \). We are done.
Problem 2: Optional (if we have enough time)
An all-paths-NFA \( M \) is a 5-tuple \( (Q, \Sigma, \delta, q_0, F) \) that accepts \( x \in \Sigma^* \) if and only if every possible state that \( M \) could be in after reading \( x \) is a state from \( F \). Prove that all-NFAs recognize exactly the regular languages. (Notice the contrast with NFAs)

Solution 2: Note, first, that an all-paths-NFA has the same syntax as an NFA, but their acceptance criteria are different. An NFA accepts a string \( x \) if and only if at least one of the computation paths leads to an accepting state. On the other hand, an all-paths-NFA accepts a string if and only if every computation path ends in an accepting state. In particular, the same machine can act as an NFA and an all-paths-NFA, depending on the acceptance criterion it uses.

Now, on to the solution: \( M \) is an all-paths-NFA. The gameplan is to show that, for every \( L \) that is accepted by some all-paths-NFA \( M \), there is an NFA \( N \) that accepts \( L \). This will prove that, if \( L \) is accepted by some all-paths-NFA, \( \bar{L} \) is regular. Since the class of regular languages is closed under complement, \( L \) is regular too.

Let \( L = L(M) \). We show that the NFA \( N = (Q, \Sigma, \delta, q_0, F' = Q - F) \), accepts \( L \). Suppose \( x \in L \). Then, by definition, all the computation paths of \( M \) on \( x \) leads to a state in \( F \). Which means none of the computation paths end in \( F' = Q - F \). Therefore, \( N \) (an NFA) does not accept \( x \), and \( x \notin L(N) \). It is not hard to see that, if \( x \notin L \) (that is, it is not accepted by \( M \)), then \( x \) is accepted by \( N \), and therefore \( x \in L(N) \). (We leave this as an exercise). We have shown that, \( x \in L(M) \) if and only if \( x \notin L(N) \). Therefore, \( N \) accepts \( \bar{L} \). ■