Solution 4.1.

DFA B is DFA A with noise, so we can modify DFA A into DFA B. We add two more states for each state and we extend each state to keep track of two things: the state identifier and the cycle they are in, the state identifier is the same as the state we have in DFA A, and the cycle is marked in 0,1 or 2, indicating whether the character is noise or not. With these extensions, state q1 in DFA A would be modified into 3 states in DFA B like this: (q1,0),(q1,1),(q1,2). The cycle marked with 0 is the state where we get character without noise, it is the same as the state in DFA A which decides if the string is character without noise. Cycle marked with 1 and 2 indicate characters with noise. The transition rule for DFA B is similar to DFA A, with the additional steps of processing noise. It is going to move to the next cycle with the same state identifier regardless of the input because it's the noise character. When moving to the next state with a different state identifier, it is checking whether the the character without noise matches. The final state will have two more states to allow 2 more additional noise characters that come after the character without noise.

Solution 4.1.

1. $Q' = Q \times \{0, 1, 2\}$
2. Σ is the same as Σ in DFA A
3. $q'_0 = (q_0, 0)$
4. δ':

   $(q_i, 0) \xrightarrow{\Sigma} (q_i, 1)$,(processing noise, move to the second position in the cycle)

   $(q_i, 1) \xrightarrow{\Sigma} (q_i, 2)$,(processing noise, move to the third position in the cycle)

   $(q_i, 2) \xrightarrow{\textit{accepted character}} (q_{i+1}, 0)$,(processing character without noise, move to the next state identifier and reset the cycle)
5. $F' : (q_f, 0),(q_f, 1),(q_f, 2)$ where $q_f \in F$