Let \( L \) be an arbitrary regular language over the alphabet \( \Sigma = \{0, 1\} \). Prove that the following languages are also regular. (You probably won’t get to all of these.)

1. \( \text{FlipOdds}(L) := \{\text{flipOdds}(w) \mid w \in L\} \), where the function \( \text{flipOdds} \) inverts every odd-indexed bit in \( w \). For example:

\[
\text{flipOdds}(000111101010101) = 1010010111111111
\]

**Solution:** Let \( M = (Q, s, A, \delta) \) be a DFA that accepts \( L \). We construct a new DFA \( M' = (Q', s', A', \delta') \) that accepts \( \text{FlipOdds}(L) \) as follows.

Intuitively, \( M' \) receives some string \( \text{flipOdds}(w) \) as input, restores every other bit to obtain \( w \), and simulates \( M \) on the restored string \( w \).

Each state \((q, \text{flip})\) of \( M' \) indicates that \( M \) is in state \( q \), and we need to flip the next input bit if \( \text{flip} = \text{TRUE} \).

\[
\begin{align*}
Q' &= Q \times \{\text{TRUE}, \text{FALSE}\} \\
s' &= (s, \text{TRUE}) \\
A' &= \\
\delta'((q, \text{flip}), a) &=
\end{align*}
\]

2. \( \text{UnflipOdd1s}(L) := \{w \in \Sigma^* \mid \text{flipOdd1s}(w) \in L\} \), where the function \( \text{flipOdd1} \) inverts every other 1 bit of its input string, starting with the first 1. For example:

\[
\text{flipOdd1s}(0000111101010101) = 00000100010001
\]

**Solution:** Let \( M = (Q, s, A, \delta) \) be a DFA that accepts \( L \). We construct a new DFA \( M' = (Q', s', A', \delta') \) that accepts \( \text{UnflipOdd1s}(L) \) as follows.

Intuitively, \( M' \) receives some string \( w \) as input, flips every other 1 bit, and simulates \( M \) on the transformed string.

Each state \((q, \text{flip})\) of \( M' \) indicates that \( M \) is in state \( q \), and we need to flip the next 1 bit of and only if \( \text{flip} = \text{TRUE} \).

\[
\begin{align*}
Q' &= Q \times \{\text{TRUE}, \text{FALSE}\} \\
s' &= (s, \text{TRUE}) \\
A' &= \\
\delta'((q, \text{flip}), a) &=
\end{align*}
\]
3. FLIPODDLs(L) := \{\text{flipOdd}s(w) \mid w \in L\}, where the function \text{flipOdd} is defined as in the previous problem.

**Solution:** Let \( M = (Q, s, A, \delta) \) be a DFA that accepts \( L \). We construct a new NFA \( M' = (Q', s', A', \delta') \) that accepts FLIPODDLs(L) as follows.

Intuitively, \( M' \) receives some string \( \text{flipOdd}s(w) \) as input, guesses which 0 bits to restore to 1s, and simulates \( M \) on the restored string \( w \). No string in FLIPODDLs(L) has two 1s in a row, so if \( M' \) ever sees 11, it rejects.

Each state \((q, \text{flip})\) of \( M' \) indicates that \( M \) is in state \( q \), and we need to flip a 0 bit before the next 1 if \( \text{flip} = \text{TRUE} \).

\[
Q' = Q \times \{\text{TRUE, FALSE}\} \\
s' = (s, \text{TRUE}) \\
A' = \\
\delta'((q, \text{flip}), a) = \]

4. FARO(L) := \{\text{faro}(w, x) \mid w, x \in L \text{ and } |w| = |x|\}, where the function faro is defined recursively as follows:

\[
\text{faro}(w, x) := \begin{cases} \\
x & \text{if } w = \varepsilon \\
a \cdot \text{faro}(x, y) & \text{if } w = ay \text{ for some } a \in \Sigma \text{ and some } y \in \Sigma^* \\
\end{cases}
\]

For example, \( \text{faro}(0001101, 1111001) = 0101011100011 \). (A "faro shuffle" splits a deck of cards into two equal piles and then perfectly interleaves them.)

**Solution:** Let \( M = (Q, s, A, \delta) \) be a DFA that accepts \( L \). We construct a DFA \( M' = (Q', s', A', \delta') \) that accepts FARO(L) as follows.

Intuitively, \( M' \) reads the string \( \text{faro}(w, x) \) as input, splits the string into the subsequences \( w \) and \( x \), and passes each of those strings to an independent copy of \( M \).

Each state \((q_1, q_2, \text{next})\) indicates that the copy of \( M \) that gets \( w \) is in state \( q_1 \), the copy of \( M \) that gets \( x \) is in state \( q_2 \), and \text{next} indicates which copy gets the next input bit.

\[
Q' = Q \times Q \times \{1, 2\} \\
s' = (s, s, 1) \\
A' = \\
\delta'((q_1, q_2, \text{next}), a) =
\]