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:

   \[
   \text{flipOdds}(0001111101010101) = 1010101111111111
   \]

   **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 $\text{flip} = \text{TRUE}$.

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

   Here I am treating 0 and 1 as synonyms for $\text{TRUE}$ and $\text{FALSE}$, respectively.

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

   \[
   \text{flipOdd1s}(0000111101010101) = 00001010001001
   \]

   **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 $\text{flipOdd1s}(L)$ 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 $\text{flip} = \text{TRUE}$.

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

   Again, I am treating 0 and 1 as synonyms for $\text{TRUE}$ and $\text{FALSE}$, respectively.
3. $\text{FLIP} \text{ODD}(L) := \{\text{flipOdd}1s(w) \mid w \in L\}$, where the function $\text{flipOdd}1$ 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 $\text{FLIP} \text{ODD}(L)$ as follows.

Intuitively, $M'$ receives some string $\text{flipOdd}1s(w)$ as input, guesses which 0 bits to restore to 1s, and simulates $M$ on the restored string $w$. No string in $\text{FLIP} \text{ODD}(L)$ 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 bit if and only if $\text{flip} = \text{TRUE}$.

$$
\begin{align*}
Q' &= Q \times \{\text{TRUE}, \text{FALSE}\} \\
s' &= (s, \text{TRUE}) \\
A' &= A \times \{\text{TRUE}, \text{FALSE}\} \\
\delta'((q, \text{FALSE}), 0) &= \{(\delta(q, 0), \text{FALSE})\} \\
\delta'((q, \text{TRUE}), 0) &= \{(\delta(q, 0), \text{TRUE}), (\delta(q, 1), \text{FALSE})\} \\
\delta'((q, \text{FALSE}), 1) &= \{(\delta(q, 1), \text{TRUE})\} \\
\delta'((q, \text{TRUE}), 1) &= \emptyset
\end{align*}
$$

The last transition indicates that we waited too long to flip a 0 to a 1, so we should kill the current execution thread.

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

$$
\text{farO}(w, x) := \begin{cases} 
  x & \text{if } w = \varepsilon \\
  a \cdot \text{farO}(x, y) & \text{if } w = ax \text{ for some } a \in \Sigma \text{ and some } y \in \Sigma^*
\end{cases}
$$

**Solution:** Let $M = (Q, s, A, \delta)$ be a DFA that accepts $L$. We construct a DFA $M' = (Q', s', A', \delta')$ that accepts $\text{FARO}(L)$ 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. Because of the constraint $|w| = |x|$, machine $M'$ can accept only if $\text{next} = 1$.

$$
\begin{align*}
Q' &= Q \times Q \times \{1, 2\} \\
s' &= (s, s, 1) \\
A' &= \{(q_1, q_2, 1) \mid q_1, q_2 \in A\} \\
\delta'((q_1, q_2, \text{next}), a) &= \begin{cases} 
  (\delta(q_1, a), q_2, 2) & \text{if } \text{next} = 1 \\
  (q_1, \delta(q_2, a), 1) & \text{if } \text{next} = 2
\end{cases}
\end{align*}
$$