1. (Sipser 1.18 f, i, and l) Give regular expressions generating the languages described in parts f, i, and l of Sipser’s exercise 1.6. For the second problem, assume that the first digit of the input is considered an odd position.

f) \{ w \mid w \text{ does not contain the substring } 110 \}

i) \{ w \mid \text{every odd position of } w \text{ is a } 1 \}

l) \{ w \mid w \text{ contains an even number of } 0s, \text{ or contains exactly two } 1s \}

2. Produce a regular expression describing the language accepted by the NFA shown below using the algorithm based on generalized nondeterministic finite automata presented in Sipser (but without showing transitions labeled by the regular expression $\emptyset$). As you execute the algorithm, remove states from the original machine in the following order: first C, then A, and finally B. Show each of the steps by drawing a diagram for each of the GNFAs produced until only 2 states remain. Remember that you should start by adding distinct start and final states to the machine. Show the final regular expression you obtain.

![NFA Diagram]

3. In class I described a transformation

$$L_{\frac{1}{2}} = \{ x \mid \text{for some } y \in \Sigma^*, xy \in L \& |x| = |y| \}$$

that can be applied to any language over the alphabet $\Sigma$. I stated that regular languages are closed under this transformation and outlined two approaches to proving this claim. Like most proofs of this sort, both approaches involved assuming we were given a DFA $D = (Q, \Sigma, \delta, s, F)$ that recognized some regular language $L$ and then showing how to construct an NFA $N$ based on $D$ that recognized $L_{\frac{1}{2}}$. I provided a precise description of how to construct the desired NFA for the first approach in class.

I described a second approach (which is summarized below) informally, but I did not show how to formally describe the NFAs imagined in this second approach. For this problem, I want you to give a formal description of how to construct NFAs that use the second approach.

In the second approach, I suggested the machine $N$ could initially guess a final state $f$ which $D$ could reach after processing all the symbols in $x$ and a guessed string $y$ whose length was equal to $x$. $N$, however, would not attempt to guess the contents of $y$ at the same time that it guesses
Instead, as the symbols of \( x \) were read one by one, the machine would guess the symbols of \( y \) one by one. While it would read the symbols of \( x \) in forward order it would guess the symbols of \( y \) in reverse order. In addition to guessing the symbols of \( y \), for each letter of \( y \) guessed the machine would also guess the state the machine \( D \) would have been in just before reading that symbol in such a way that its guess was consistent with the state \( f \) and the symbols of \( y \) it had already guessed. Basically, the machine would simulate the execution of \( M \) running backward on \( y \) at the same time it simulates the execution of \( D \) running forward on \( x \).

To verify that all of its guessing is correct, the machine must be designed to check that the two simulations of \( D \) end up in the same state.

For any language \( L \) over alphabet \( \Sigma \), consider the derived languages

\[
\begin{align*}
L_{−\frac{1}{3}} &= \{ w \mid \text{for some } x, y \in \Sigma^*, |x| = |y| = |w| \text{ and } xyw \in L \} \\
L_{−\frac{1}{2}} &= \{ w \mid \text{for some } x, y \in \Sigma^*, |x| = |y| = |w| \text{ and } xwy \in L \} \\
L_{\frac{1}{4}−\frac{1}{3}} &= \{ w \mid \text{for some } x, y, z \in \Sigma^*, |x| = |y| = |z|, w = xz \text{ and } xyz \in L \}
\end{align*}
\]

Regular languages are closed under at least one of these three transformations. At the same time, regular languages are not closed under at least one of these transformations. Identify one of the transformations that preserves regularity and justify your claim that regular languages are closed under this transformation by providing a formal description of how to transform a DFA, \( M \), such that \( L = L(M) \) into a new automaton (probably a NFA) \( M' \) that recognized the transformed language \((L_{−\frac{1}{3}}, L_{−\frac{1}{2}}, \text{ or } L_{\frac{1}{4}−\frac{1}{3}})\). Provide a brief but complete explanation of the intuition behind your construction.