Exercise 1

In our proof that the regular languages are closed under the Kleene star operator – that is, if $L$ is regular, $L^*$ is regular – we used the following construction:\footnote{We also changed the existing accept states of $N$ to be regular, non-accept states. This isn’t strictly necessary, but it’s convenient if the resulting NFA needs to be joined with another, e.g., to concatenate languages, for there to be just one accept state rather than several.}

1. Begin with an NFA $N$ where $L(N) = L$.
2. Add a new start state $q_{\text{start}}$.
3. Add an $\varepsilon$-transition from $q_{\text{start}}$ to the start state of $N$.
4. Add $\varepsilon$-transitions from each accepting state of $N$ to $q_{\text{start}}$.
5. Make $q_{\text{start}}$ an accepting state.

You might have wondered why we needed to add $q_{\text{start}}$ as a new state to the NFA. It might have seemed more natural to do the following:

1. Begin with an NFA $N$ where $L(N) = L$.
2. Add $\varepsilon$-transitions from each accepting state of $N$ to the start state of $N$.
3. Make the start state of $N$ an accepting state.

Unfortunately, this construction does not work correctly.

Find a regular language $L$ and an NFA $N$ for $L$ such that using the second construction does not create an NFA for $L^*$. Justify why the language of the new NFA isn’t $L^*$. 
Exercise 2

The state elimination algorithm gives a way to transform a finite automaton into a regular expression. It’s an elegant algorithm once you get the hang of it, so for this problem you’ll try applying it to an example.

Let $\Sigma = \{a, b\}$, and let $L_2 = \{w \in \Sigma^* \mid w$ has an even number of $a$s and $b$s$\}$. Below is a finite automaton for $L_2$ that we’ve prepared for the state elimination algorithm by adding a new start state $q_{\text{start}}$ and a new accept state $q_{\text{acc}}$:

```
q_{\text{acc}}
```

\[ q_{\text{start}} \rightarrow \epsilon \quad \epsilon \rightarrow q_0 \]

\[ q_0 \rightarrow a, b \quad a \rightarrow q_1, b \rightarrow q_2 \]

\[ q_1 \rightarrow a, b \quad a \rightarrow q_2, b \rightarrow q_3 \]

\[ q_2 \rightarrow a, b \quad a \rightarrow q_0, b \rightarrow q_1 \]

\[ q_3 \rightarrow a, b \quad a \rightarrow q_2, b \rightarrow q_3 \]

a. Follow the state elimination algorithm to remove $q_1$ and then $q_2$. Show your result at this point.

b. Finish the state elimination algorithm by removing $q_3$ and then $q_0$, showing your work.
Exercise 3

Let \( \Sigma = \{ (, ) \} \). Consider the following language over \( \Sigma \):

\[
L_3 = \{ w \in \Sigma^* \mid w \text{ is a string of balanced parentheses} \}
\]

For example,

\[
( ) \in L_3 \quad ( ( ) ) \in L_3 \quad ) ( \notin L_3 \\
( ( ) ( ) ) \in L_3 \quad ( ( ) ) \notin L_3 \\
( ( ) ) ( ( ) ) \in L_3 \quad ( ) ) ) \notin L_3 \\
\epsilon \in L_3
\]

Use the Pumping Lemma to prove that \( L_3 \) is non-regular.

I realize it can be confusing to write about the symbols \( ( \) and \( ) \). Feel free to substitute \( l \) and \( r \) or \( a \) and \( b \) if you prefer.
Exercise 4

Suppose we want to check whether $x + y = z$, where $x$, $y$, and $z$ are all natural numbers. We can formulate this problem as a language using the unary number system, where the number $n$ is represented by writing out $n$ 1s.² For example, the number 5 would be written as 11111. Then, given the alphabet $\Sigma = \{1, \ +, =\}$, we can encode $x + y = z$ by writing out $x$, $y$, and $z$ in unary. For example:

- $3 + 4 = 7$ would be encoded as $111 + 1111 = 1111111$
- $7 + 1 = 8$ would be encoded as $1111111 + 1 = 11111111$
- $0 + 1 = 1$ would be encoded as $+1 = 1$

Consider the language $ADD = \{1^m + 1^n = 1^{m+n} \mid m, n \in \mathbb{N}\}$. That is, $ADD$ consists of strings encoding two unary numbers and their sum. Use the Pumping Lemma to prove that $ADD$ is not a regular language.

² You can think of this like tally marks.