1. **(SOLO)** Which of the following languages are regular? If you think a language is regular, draw the state diagram of a DFA that recognizes the language. You do not have to prove that the DFA recognizes the language. If you think a language is not regular, than you have to provide a proof.

(a) \( L = \{ w \in \{0, 1\}^* | \text{w contains at exactly three 1's} \} \)

(b) \( L = \{ a^n b^m | n, m > 0, n \equiv m \mod 3 \} \), where \( \Sigma = \{a, b\} \).

(c) \( L = \{ xay : x, y \in \{a, b\}^*, |x| = |y| \} \), where \( \Sigma = \{a, b\} \). \( (L \text{ is the set of strings with an a at the center.}) \)

(d) \( L = \{ xwx^R | x, w \in \Sigma^*, |x|, |w| > 0 \} \), where \( \Sigma = \{a, b\} \).

(e) \( L = \{ w0^n w : w \in \Sigma^*, |w| > 0, n > 0 \} \), where \( \Sigma = \{0, 1\} \).

2. **(SOLO)** DFAs cannot count but they can verify addition. Consider an alphabet with rather interesting symbols:

\[ \Sigma = \left\{ \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right\} \]

These symbols represent columns which could appear when writing the addition and result of two binary numbers. An input of length \( n \) represents the addition of two \( n \)-bit numbers where the input is given from least to most significant bit. The first number’s binary representation is in the top row, the second number’s representation is in the middle row, and the result is in the bottom row. Let \( L \) be the language of strings over this alphabet that represent correct addition statements as described above. As an example, correctly representing \( 3 + 4 = 7 \) in this scheme would give the following string (remember that it is ordered least significant bit to most, so the first input symbol is on the left):

\[
\begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}
\]

The following represents \( 5 + 2 = 6 \), and should therefore not be in \( L \):

\[
\begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}
\]

Draw a DFA which accepts \( L \). Try not to use more than 5 states.
3. **(SOLO)**) In this question we will define a new computing machine: DFAs with magical superpowers. We'll call them MFAs. Just like a DFA, an MFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$. The only difference is that in an MFA, the transition function $\delta$ has the form $\delta : Q \times \Sigma \to \mathcal{P}(Q)$, where $\mathcal{P}(Q)$ denotes the powerset of $Q$ (which includes the empty set). We say that an MFA $(Q, \Sigma, \delta, q_0, F)$ accepts a string $w = a_1a_2\ldots a_n$ if there exists a sequence of states $(r_0, r_1, r_2, \ldots, r_n)$ such that:

(i) $r_0 = q_0$, i.e., $r_0$ is the initial state,
(ii) $r_i \in \delta(r_{i-1}, a_i)$, for $i \in \{1, 2, \ldots, n\}$,
(iii) $r_n \in F$.

We can draw an MFA similar to how we draw a DFA. For example, the following is an MFA:

![MFA Diagram]

Note that there are a few differences from a DFA. Given a state, there might be multiple edges going out of it corresponding to the same symbol. For example, $q_0$ has two edges going out of it labeled with 1. Also, there is not necessarily an edge going out of a state that corresponds to every possible symbol. For example, there is no edge going out of $q_1$ with the label 1. Let’s say we get the string 101 as input. There are multiple paths we can take when we read this string since at $q_0$ we have two options when reading a 1. If one of these possible paths ends in an accepting state, then the MFA accepts the string. And otherwise, the MFA rejects the string. In the case of 101, the string is accepted because the sequence $(q_0, q_0, q_0, q_1)$ is one of the allowed paths, and it ends in an accepting state. (Convince yourself that this verbal description of what it means for an MFA to accept a string corresponds exactly to the symbolically precise definition given above with the three parts (i), (ii), and (iii).)

We will now extend the definition of a DFA even more. Define an SFA to be a machine that is just like an MFA, but it is allowed to have transitions (edges) labeled with the $\varepsilon$ symbol. For example, the following is a drawing of an SFA:
When processing a string, if we are at some state \( q \), and this state has an edge labeled with \( \epsilon \), then we are allowed to follow that edge and go to the next state \textbf{without} consuming any input symbol. (In the example above, the \( \epsilon \) transitions basically allow us to start at state \( q_1 \) or state \( q_3 \).) As in an MFA, a string is accepted if there is some path we can follow that ends in an accepting state. For example, the string 00 is accepted by the SFA above because we can first take the \( \epsilon \) transition to go to state \( q_3 \) (without consuming the first symbol of the string), and then read a 0 symbol to stay at \( q_3 \), and finally read a 0 symbol again to go from \( q_3 \) to \( q_4 \).

(a) Give a more symbolically precise definition of what it means for an SFA to accept a string \( w \), akin to the (i),(ii),(iii) description above for MFAs.

(b) Give an MFA or SFA for the language \( A_{123} = \{ x \in \{0,1\}^* \mid x \text{ has 1111011 as a substring} \} \).

(c) Give a DFA for \( A_{123} \). Did you use any extra states compared to your MFA/SFA from part (i)?

(d) Given an MFA for \( B_{R6} = \{ x \in \{0,1\}^* \mid \text{the 6th bit of } x \text{ from the right is a 1} \} \). Your MFA must have at most 7 states.

(e) Prove that any DFA for \( B_{R6} \) must have at least 64 states.

4. \textbf{(GROUP)} Show that a language recognized by an SFA is also a regular language.

\textit{Suggestion: As a warm-up, you can show that a language recognized by any MFA is a regular language.}

5. \textbf{(GROUP)} Suppose that \( A \subseteq \Sigma^* \) is a regular language. Show that

\[ \{ x \in \Sigma^* \mid \exists y \in \Sigma^* \text{ such that } xy \in A \text{ and } |x| = |y| \} \]

is a regular language.

\textit{Note: For this problem, you are allowed to build an MFA or SFA to show regularity.}

6. \textbf{(OPEN)} Suppose \( A \subseteq \Sigma^* \) is an arbitrary regular language. True or False: the language \( \{ x \in \Sigma^* \mid xx \in A \} \) is also regular. Justify your answer — if you answer False, give a counterexample, and if you answer True, argue why the language is regular (eg. by giving a DFA, or SFA if that helps).

7. \textbf{(SOLO)}

(a) Suppose \( M = (Q, \Sigma, \delta, q_0, F) \) is a DFA accepting a language \( A \subseteq \Sigma^* \). Give a Turing Machine \( N \) that accepts precisely the strings in \( A \) and rejects those in \( \Sigma^* \setminus A \). (You can either informally describe how the state diagram of \( N \) relates to that of \( M \), or give the formal 7-tuple defining \( N \).)

(b) Turing Machines can move around data on its tape in various ways by using its state space and marking of tape cells to guide it. As a simple example, it can shift the entire input string one cell to the right, by moving the right-most symbol over, then next one, and so on, operating as follows: \textit{“Move to the end of the string, remembering the last symbol seen as you go. When you reach the right-hand end, write down the remembered symbol, then back up and erase the symbol. Then repeat...”}

Draw the state transition diagram of a Turing Machine that operates as above and shifts the entire input string one cell to the right.