Problem 1

Answer. The NFA \( M = (Q, \Sigma, \delta, p_0, F) \) is as follows.

\[
\begin{array}{c}
0 \\
p_0 \\
\epsilon \\
p_1 \\
1 \\
p_2 \\
0
\end{array}
\]

Here \( Q = \{p_0, p_1, p_2\} \), \( \Sigma = \{0, 1\} \).

The transition function \( \delta \) is as follows.

\- \( \delta(p_0, 0) = \{p_0\}, \delta(p_0, 1) = \emptyset, \delta(p_0, \epsilon) = \{p_1\} \).
\- \( \delta(p_1, 0) = \{p_2\}, \delta(p_1, 1) = \{p_1\}, \delta(p_1, \epsilon) = \emptyset \).
\- \( \delta(p_2, 0) = \{p_2\}, \delta(p_2, 1) = \emptyset, \delta(p_2, \epsilon) = \emptyset \).

The start state is \( p_0 \).

The set of accept states is \( F = \{p_2\} \).

Problem 2

Answer. As \( A \) is regular, there is a DFA \( M' = (Q, \Sigma, \delta, p_0, F') \) that recognizes NOEXTEND\((A)\). Notice that \( M' \) and \( M \) are only different in their sets of accept states.

We define \( F' \subseteq F \) as follows. For every state \( p \in F \), \( p \) is in \( F' \) if and only if \( \forall q \in F \) (including the case \( q = p \)), there is no path from \( p \) to \( q \) in the diagram of \( M \).

Next we prove that \( M' \) recognizes NOEXTEND\((A)\).

For every string \( x \in \text{NOEXTEND}(A) \), according to the definition of NOEXTEND\((A)\), we know \( x \in A \). By reading \( x \), starting from \( p_0 \), \( M \) ends at a state \( u \in F \). As \( x \) is not the proper prefix of any string in \( A \), there is no path from \( u \) to any \( q \in F \) in the diagram of \( M \). So \( u \in F' \). As a result, \( x \) is accepted by \( M' \).

For every string \( x \) that is accepted by \( M' \). Starting from \( p_0 \), following \( x \), \( M' \) ends at a state \( u \in F' \). Since \( F' \subseteq F \), we have \( x \in A \). Suppose \( x \) is a proper prefix of a string \( y \in A \). Assume \( y = x \circ z \). As \( y \in A \), starting from \( u \), following \( z \), \( M \) ends at a state \( q \in F \). However this contradicts the definition of \( F' \). So \( x \) is not a proper prefix of any string in \( A \). So \( x \in \text{NOEXTEND}(A) \).

This proves that \( M' \) recognizes NOEXTEND\((A)\).
Problem 3

Answer. According to the definition of regular languages, it is sufficient to show that there exists a DFA $M$ that recognizes $C_n$.

Let $M = (Q, \Sigma, \delta, q_0, F)$. Here $Q = \{0, 1, 2, \ldots, n - 1, q_0\}$. Each state except $q_0$ in $Q$ corresponds to a possible residue of $x \mod n$. As there are $n$ different residues for $x \mod n$, so the number of states is $n + 1$.

Let $\Sigma = \{0, 1\}$.

The start state is $q_0$.

For the transition function, $\forall q \in Q - \{q_0\}$ and $\forall c \in \Sigma$, we have $\delta(q, c) = (2q + c) \mod n$. Also $\forall c \in \Sigma, \delta(q_0, c) = c \mod n$.

The accept state is the 0 state.

Next we prove that $M$ recognizes $C_n$.

For an input $x$, if it is a multiple of $n$, then we can find a corresponding path $P$ in the graph of $M$. This path corresponds to a procedure of doing the modular operation and the final state is exactly the result of $x \mod n$. The first state of $P$ is $q_0$. The $i$th state in $P$ is $\bar{x}_{i-1} \mod n$, for $i = 2, \ldots, |x| + 1$, where $\bar{x}_i$ is the first $i$ bits of $x$. $|x|$ is the length of $x$. So $P$ finally reaches the accept state 0.

For a string $x$ accepted by $M$, there is a path $P$ corresponding to it. We know that $P$ ends at an accept state. As $P$ corresponds to a procedure of computing $x \mod n$ and the residue is 0, $x$ is in $C_n$.

This proves that $M$ recognizes $C_n$. 

\[\square\]

Problem 4

Answer. We construct a DFA $M$ such that the language $L$, which is the perfect shuffle of $A$ and $B$, is recognized by $M$.

Assume the DFA that recognizes $A$ is $M_A = (Q_A, \Sigma, \delta_A, q_{A0}, F_A)$. The DFA that recognizes $B$ is $M_B = (Q_B, \Sigma, \delta_B, q_{B0}, F_B)$.

Let $M = (Q, \Sigma, \delta, q_0, F)$.

Each state of $M$ is represented by a 3-tuple $(x, y, z)$, where $x$ is a state of $M_A$. $y$ is a state of $M_B$ and $z$ is a binary bit. Let $z = 0$ indicate that $M$ is supposed to read an $a_i$. Let $z = 1$ indicate that $M$ is supposed to read a $b_i$. So $Q = Q_A \times Q_B \times \{0, 1\}$.

$\Sigma$ is exactly the $\Sigma$ given in the description of the problem.

At any state $(x, y, 0)$, consider the state $x$ of $M_A$. For each outgoing edge of $x$, we construct a corresponding outgoing edge of $(x, y, 0)$. If $\delta_A(x, a_e) = x'$, then $\delta((x, y, 0), a_e) = (x', y, 1)$. That is, $\delta((x, y, 0), a_e) = (\delta_A(x, a_e), y, 1)$.

At any state $(x, y, 1)$, consider the state $y$ of $M_B$. For each outgoing edge of $y$, we construct a corresponding outgoing edge of $(x, y, 1)$. If $\delta_B(y, b_e) = y'$, then $\delta((x, y, 1), b_e) = (x, y', 0)$. That is, $\delta((x, y, 1), b_e) = (x, \delta_B(y, b_e), 0)$.

The start state is $q_0 = (q_{A0}, q_{B0}, 0)$.

The accept states are the states $(x, y, 0), \forall x \in F_A, \forall y \in F_B$. That is, $F = F_A \times F_B \times \{0\}$.

For any $w = a_1b_1 \cdots a_kb_k \in L$, we can find a path $P$ in $M$ composed of the series of edges: $a_1b_1 \cdots a_kb_k$. The final state of $P$ is an accept state. The reason is as follows. Assume that by reading $w$, $M$ ends at a state $(x, y, 0)$. We know that $a_1a_2 \cdots a_k \in A$ and $b_2b_2 \cdots b_k \in B$. This means that by reading $a_1a_2 \cdots a_k$, $M_A$ ends at a state in $F_A$. Also by reading $b_1b_2 \cdots b_k$, $M_B$ ends at a state in $F_B$. So according to our construction, we have $x \in F_A$ and $y \in F_B$. So $(x, y, 0) \in F$. 

2
On the other hand, if $P$ is a path in $M$ from the start state to an accept state, assuming the $i$th edge is $P_i$, $i = 1, 2, \ldots, 2k$, we know from the construction that if $i$ is odd, then $P_i$ corresponds to an edge in $M_A$. What’s more, $P_1, P_3, \ldots, P_k$ form a path which starts at the start state of $M_A$ and ends at an accept state of $M_A$. If $i$ is even, we can find out that $P_2, P_4, \ldots, P_{2k}$ form a path which starts at the start state of $M_B$ and ends at an accept state of $M_B$. So the string corresponding to $P$ is in $L$.

So the perfect shuffle of $A$ and $B$ is recognized by $M$. Thus the class of regular languages is closed under perfect shuffle.

\[\square\]

**Problem 5**

**Answer.** Assume the DFA that recognizes $A$ is $M_A = (Q_A, \Sigma, \delta_A, q_{A0}, F_A)$. The DFA that recognizes $B$ is $M_B = (Q_B, \Sigma, \delta_B, q_{B0}, F_B)$.

Next we construct an NFA $M$ that recognizes the shuffle of $A$ and $B$. Let $M = (Q, \Sigma, \delta, q_0, F)$.

Let $Q = Q_A \times Q_B$.

For every $x \in Q_A$, every $y \in Q_B$ and every $c \in \Sigma$, $\delta((x, y), c) = \{(\delta_A(x, c), y), (x, \delta_B(y, c))\}$. For every $x \in Q_A$, every $y \in Q_B$, $\delta((x, y), \epsilon) = \emptyset$.

Let $q_0 = (q_{A0}, q_{B0})$.

Let $F = F_A \times F_B$.

For any $w = a_1b_1\cdots a_kb_k, a_1\cdots a_k \in A$ and $b_1\cdots b_k \in B, a_i, b_i \in \Sigma^*$, there is a path in $M$ starting from the start state and ends at an accept state. The path is $a_i, b_i, \cdots, a_k, b_k$. The path ends at an accept state because for the entire string $a_1a_2\cdots a_k, M$ uses the first transition rule (according to $\delta_A$) and for the entire string $b_1b_2\cdots b_k$, $M$ uses the second transition rule (according to $\delta_B$). So $w$ is accepted by $M$.

For any string $w$ that is accepted by $M$, assume $w$ corresponds to a path $P$ in $M$. Assume the final state of $P$ is $(q_1, q_2) \in F_A \times F_B$. We construct the following two strings $a$ and $b$. At first, let $a = b = \epsilon$. For every $i = 1, 2, \ldots, |w|$, if the edge $w_i$ in $P$ goes from $(x, y)$ to $(x', y)$ such that $\delta_A(x, w_i) = x'$, then we pad $w_i$ to $a$; else we pad $w_i$ to $b$. As a result, we can see that starting from $q_{A0}$, following $a$, $M_A$ ends at the state $q_1 \in F_A$. Also starting from $q_{B0}$, following $b$, $M_B$ ends at the state $q_2 \in F_B$. So $w$ is in the shuffle of $A$ and $B$.

So $M$ recognizes the shuffle of $A$ and $B$. Thus the class of regular languages is closed under shuffle.

\[\square\]