1. Construct nondeterministic finite automata for the following regular expressions (answers not shown).

   (a) \((aba)^*\)
   
   (b) \((ab \cup a)^*\)

2. Construct regular expression for \(L = \{w \mid w \text{ does not contain the substring 010}\}\) where \(\sum = \{0, 1\}\).

   Answer:
   You can either write a regular expression right off from the language definition, or construct a finite automaton first and convert it to a regular expression.
   Let us assume that we have constructed the following DFA.

   ![DFA Diagram]

   Then, for each state, try to list possible inputs that would result in that state. The final result (one of many possible solutions) is \((1^*0^+11)^*(1^*(\varepsilon \cup 0^+ (\varepsilon \cup 1)))\).

   In Ruby, we can write the regular expression as \(/(1*0+11)*(1*(0+1?))/\).

   Note: This is just one of many possible answers, and the conversion of the DFA to the regular expression is done informally. Refer to other texts for more detail about this conversion if you’d like to know more on this topic. This not discussed in CMSC330).

3. Convert the regular expression to DFA:
   
   \(0^*(1^*0 \cup 0)0^*\)
Answer: (Again, this is only an instance of many possible answers)
First, construct a nondeterministic finite automaton for the regular expression (which is fairly easy). You could simplify the regular expression or the NFA first, but we want to show how to convert NFA to DFA, so we will use the NFA shown below.

Then, construct an NFA whose states are the superset of the states in the original NFA, which should include an empty set state $\emptyset$. Make sure to get transitions (edges) correct since it might be tricky to get it right first time. (Keep in mind that we must follow $\epsilon$ in every transition.)

Finally, remove the unnecessary states.

4. Prove that the class of regular language is closed under the union operation.

Answer:
If we can construct a new NFA that recognizes a union of two languages recognized by two NFA, this proves that the language is closed under the union operation.
Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize $A_2$.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

(a) $Q = \{q_0\} \cup Q_1 \cup Q_2$. The states of $N$ are all the states $N_1$ and $N_2$, with the addition of a new state $q_0$.

(b) The state $q_0$ is the start state of $N$.

(c) The accept states $F = F_1 \cup F_2$. The accept states of $N$ are all the accept states of $N_1$ and $N_2$. The way $N$ accepts if either $N_1$ accepts or $N_2$ accepts.

(d) Define $\delta$ so that for any $q \in Q$ and any $a \in \Sigma^*$,

$$
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & q \in Q_1 \\
\delta_2(q, a) & q \in Q_2 \\
\{q_1, q_2\} & q = q_0 \text{ and } a = \epsilon \\
\emptyset & q = q_0 \text{ and } a \neq \epsilon 
\end{cases}
$$

5. Prove that the class of regular language is closed under the intersection operation.

Answer:

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize $A_1$, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize $A_2$. Now construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cap A_2$.

(a) $Q = Q_1 \times Q_2$. The states of $N$ are Cartesian product of $N_1$ and $N_2$. 
(b) The state \( q_0 \) is the start state of \( N \) where \( q_0 = (q_1, q_2) \).

(c) The accept states \( F = \{(q_a, q_b) \mid q_a \in F_1, q_b \in F_2 \} \). The accept states of \( N \) are intersection of the accept states of \( N_1 \) and \( N_2 \).

(d) Define \( \delta \) such that \( \delta((q_n, q_m)) = (\delta_1(q_n), \delta_2(q_m)) \).

Note: This proof works for DFA. You might have to refine this proof in order to make it work for NFA. Try it.

Also, you can use the DeMorgan’s Law instead—i.e., construct a finite automaton that recognizes \( \overline{A_1 \cup A_2} \). This might be easier, but you have to prove that the class of regular language is closed under the union operation and the negation operation (complement).