Problem 1.

Consider the following $\epsilon$-NFA.

<table>
<thead>
<tr>
<th></th>
<th>$\epsilon$</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rightarrow p$</td>
<td>${q, r}$</td>
<td>$\emptyset$</td>
<td>${r}$</td>
<td>${q}$</td>
</tr>
<tr>
<td>$q$</td>
<td>$\emptyset$</td>
<td>${p}$</td>
<td>${p, q}$</td>
<td>${r}$</td>
</tr>
<tr>
<td>$\rightarrow r$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

(a) Compute the $\epsilon$-closure of each state.
(b) Give all the strings of length three or less accepted by the automaton.
(c) Convert the automaton to a DFA.

(a)

$$\text{ECLOSE}(p) = \{p, q, r\}$$

$$\text{ECLOSE}(q) = \{q\}$$

$$\text{ECLOSE}(r) = \{r\}$$

(b) First we check $\epsilon$, $a$, $b$, and $c$ and they’re all accepted. In addition, we have $p \in \delta^* (p, a)$ and $p \in \delta^* (p, b)$ which means if the first bit is $a$ or $b$, we can choose to go back to start state and start over. Hence, $a(a + b + c)$ and $b(a + b + c)$ are all accepted. Now we check $ca$, $cb$, $cc$, and they’re all accepted.

Now we know $(a + b + c)(a + b + c)$ (all strings of length 2) are all accepted. For the same reason as above, we have $a(a + b + c)(a + b + c)$ and $b(a + b + c)(a + b + c)$ be accepted. However, there are no ways to accept $cca$, $ccb$, or $ccc$. 

1
Here, we list all accepted strings of length 3 or less:

\[
\begin{align*}
\epsilon, & \ a, b, c, \\
aa, & \ ab, & \ ac, & \ ba, & \ bb, & \ bc, & \ ca, & \ cb, & \ cc, \\
\text{aaa}, & \ aab, & \ aac, & \ aba, & \ abb, & \ abc, & \ aca, & \ acb, & \ acc, \\
\text{baa}, & \ bab, & \ bac, & \ bba, & \bbb, & \ bbc, & \ bca, & \ bcb, & \ bcc, \\
\text{caa}, & \ cab, & \ cac, & \ cba, & \ cbb, & \ cbc,
\end{align*}
\]

\( \star \{ p, q, r \} \) 
\( \star \{ q, r \} \) 
\( \star \{ r \} \) 
\( \emptyset \) 

Note that since \( \text{ECLOSE}(p) = \{ p, q, r \} \), the start state is \( \{ p, q, r \} \) instead of \( \{ p \} \).
Problem 2.

Write a regular expression for the following language: The set of all binary strings whose number of 0's is divisible by four.

\[ 1^*(01^*01^*01^*)^* \]
**Problem 3.** (Exercise 3.2.3)

Convert the following DFA to a regular expression, using the state-elimination technique of Section 3.2.2.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>→</td>
<td>*p</td>
<td>s</td>
</tr>
<tr>
<td></td>
<td>q</td>
<td>p</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>s</td>
<td>q</td>
<td>r</td>
</tr>
</tbody>
</table>

For the convenience of grading, please eliminate states in the order q, r, s.

Original NFA:

After eliminating state q:

After eliminating state r:
After eliminating state $s$:

$$1 + 0(01 + 10^{*}11)^{*}(00 + 10^{*}10)$$

The regular expression is $(1 + 0(01 + 10^{*}11)^{*}(00 + 10^{*}10))^*$. 
Problem 4. (Exercise 3.2.6(c)(d))

Let $A = (Q, \Sigma, \delta, q_0, \{q_f\})$ be an $\epsilon$-NFA’s such that there are no transitions into $q_0$ and no transitions out of $q_f$. Describe the language accepted by each of the following modifications of $A$, in terms of $L = L(A)$: (do (c) and (d))

(b) The automaton constructed from $A$ by adding an $\epsilon$-transition from $q_0$ to every state reachable from $q_0$ (along a path whose labels may include symbols of $\Sigma$ as well as $\epsilon$).

(c) The automaton constructed from $A$ by adding an $\epsilon$-transition to $q_f$ from every state that can reach $q_f$ along some path.

(d) The automaton constructed from $A$ by doing both (b) and (c).

(You may describe the languages using a combination of plain English and mathematics notations (such as set notations). Please also consult the answers for parts (a) and (b) available on the textbook homepage: http://www-db.stanford.edu/~ullman/ialc.html)

(c) The set of prefixes of strings in $L$, including $\epsilon$ if $L \neq \emptyset$.

(d) The set of substrings of strings in $L$, including $\epsilon$ if $L \neq \emptyset$. 

6
**Problem 5.** (Exercise 3.4.1(c)(d))

Verify the following identities involving regular expressions.

(c) \((RS)T = R(ST)\).

(d) \(R(S + T) = RS + RT\).

(You may use the test technique in Section 3.4.7 (also see the sample solution for part a) on the textbook [homepage](#)), or the general proof technique used in the proof of Theorem 3.11. With the test technique, we argue that both sides represent the same language \(\{abc\}\) for part (c) and \(\{ab,ac\}\) for part (d).)

(c) \((\Rightarrow)\) Suppose \(w\) is in \((RS)T\), let \(w = uz\) such that \(u\) is in \(RS\) and \(z\) is in \(T\). Further we let \(u = xy\) such that \(x\) is in \(R\) and \(y\) is in \(S\). Now \(w = xyz = x(yz)\), \(x\) is in \(R\), \(yz\) is in \(ST\), so \(w\) is in \(R(ST)\).

\((\Leftarrow)\) Likewise.

(d) \((\Rightarrow)\) Suppose \(w\) is in \(R(S + T)\), let \(w = xy\) such that \(x\) is in \(R\) and \(y\) is in either \(S\) or \(T\). If \(y\) is in \(S\), then \(xy\) is in \(RS \subseteq RS + RT\). Otherwise \(xy\) is in \(RT\), then \(xy\) is in \(RT \subseteq RS + RT\).

\((\Leftarrow)\) Suppose \(w\) is in \(RS + RT\), then \(w\) is in either \(RS\) or \(RT\). If \(w\) is in \(RS\), let \(w = uv\) such that \(u\) is in \(R\) and \(v\) is in \(S\). Since \(S \subseteq S + T\), \(v\) is in \(S + T\), \(w = uv\) is in \(R(S + T)\). If \(w\) is in \(RT\), likewise.