1. Convert the regular expression \((ab \cup c)^*\) into an NFA using the method from class (which is the same as the method from the book and is different from the method in JFLAP).

**Solution:**

First we have the following NFAs for \(a\), \(b\), and \(c\).

\(a:\) 

\[ 
\text{\(\epsilon\)} \quad \text{\(a\)} \quad \text{\(\epsilon\)} 
\]

\(b:\) 

\[ 
\text{\(\epsilon\)} \quad \text{\(b\)} \quad \text{\(\epsilon\)} 
\]

\(c:\) 

\[ 
\text{\(\epsilon\)} \quad \text{\(c\)} \quad \text{\(\epsilon\)} 
\]

Then, we obtain an NFA for \(ab\)

\[ 
\text{\(\epsilon\)} \quad \text{\(a\)} \text{\(\epsilon\)} \quad \text{\(\epsilon\)} \quad \text{\(b\)} \quad \text{\(\epsilon\)} 
\]

and the following for \((ab \cup c)\).

\[ 
\text{\(\epsilon\)} \quad \text{\(a\)} \quad \text{\(\epsilon\)} \quad \text{\(b\)} \quad \text{\(\epsilon\)} 
\]

\[ 
\text{\(\epsilon\)} \quad \text{\(c\)} \quad \text{\(\epsilon\)} 
\]
Finally, we obtain the NFA below for \((ab \cup c)^*\).

2. Let \(N\) be the following NFA:
Convert $N$ into a regular expression using the method from class (which is the same as the method in the book, and is not the same as the method in JFLAP).

**Solution:**

Normally, when we turn an NFA into a GNFA, we have to add a new start state with an $\varepsilon$-transition to the old start state, but in this case, the NFA has no transitions into the start state, so it is not necessary to add a new start state, so we get the following GNFA when we transform the NFA to a GNFA.
Eliminating state $q_1$ gives

Eliminating state $q_2$ gives

Eliminating state $q_3$ gives
Finally, eliminating state $q_4$ gives

and the regular expression is $((a^* b \cup \varepsilon)a^* a \cup a^* b^*)^* (ba^* a) \cup a^* b^*$.

3. **Problem 1.31** Show that if $A$ is regular so is $A^R$.

**Proof:** Since $A$ is regular, there is some NFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A$. We construct an NFA $M'$ such that $L(M') = A^R$. The idea is that the start state of $M'$ is the accept state of $M$, the accept state of $M'$ is the start state of $M$, and for every transition in $M$, there is a transition with the same label in $M'$, but going in the reverse direction. The problem with this is that $M$ may have more than one accept state. Since $M'$ can have only one start state, we add a new start state to $M'$. 
and $\varepsilon$ transitions from the new start state to all the accept states of $M$.
The formal construction is:

$$
M' = (Q \cup \{q_s\}, \Sigma, \delta', q_s, \{q_0\})
$$

where $q_s$ is a new state and

$$
\delta'(q, a) = \begin{cases}
\{p : q \in \delta(p, a)\} & \text{if } q \neq q_s \\
F & \text{if } q = q_s, a = \varepsilon \\
\emptyset & \text{if } q = q_s, a \neq \varepsilon
\end{cases}
$$

4. (a) Let $M$ be the NFA given in the solution to Problem 2(b) on Homework 2. Give an NFA $N$ with two states and no $\varepsilon$-transitions that recognizes the same language. (Your NFA can have more than one accept state.)

**Solution:**

```
   0
  ↓  ↓
  ◯   ◯
  1  1,2
```

(b) Generalize what you did in Part (a) of this problem by proving the following theorem:

**Theorem:** If $M$ is an NFA, then there is an NFA $N$ with the following properties

1. $N$ has the same number of states as $M$.
2. $N$ has no $\varepsilon$-transitions.
3. $L(N) = L(M)$.

**Solution:** We obtain $N$ from $M$ by doing three things:

1. We remove all $\varepsilon$-transitions.
2. If in $M$ there is a series of $\varepsilon$-transitions that leads from a state $p$ to an accept state $q$, then in $N$, $p$ is also an accept state.
3. If in $M$ there is a series of $\varepsilon$-transitions from a state $p$ to a state $q$, and $M$ can go from $q$ to a state $r$ reading a symbol $a$, then $N$ can go directly from $p$ to $r$ reading $a$.

Formally, let $M = (Q, \Sigma, \delta, q_0, F)$. We define $N = (Q, \Sigma, \delta', q_0, F')$. If $q$ is a state in $Q$, we use the notation $E_M(q)$ for the set of states that can be reached from $q$ using 0 or more $\varepsilon$-transitions in $M$. The definition of $N$ is now given by
5. Use the Pumping Lemma to show that the following languages are not regular:

(a) \( \{a^nb^mc^n | n, m \geq 0 \text{ and } r = n + m \} \):

Solution: Given \( p \geq 1 \), choose \( s = a^pb^pc^{2p} \). Then, \( s \) is in the language and \( |s| = 4p \geq p \). Given \( x, y, z \) with \( s = xyz \), \( |xy| \leq p \) and \( |y| > 0 \), we choose \( i = 2 \). Then, since \( |xy| \leq p \), \( y \) consists only of \( a \)'s, so \( xy^2z = a^{p+|y|}b^pc^{2p} \) which is not in the language since \( |y| > 0 \), so \( p + |y| + p \neq 2p \).

(b) \( \{0^n10^m | n \leq m \} \).

Solution: Given \( p \geq 1 \), choose \( s = 0^p10^p \). Then, \( s \) is in the language and \( |s| = 2p + 1 \geq p \). Given \( x, y, z \) with \( s = xyz \), \( |xy| \leq p \) and \( |y| > 0 \), we choose \( i = 2 \). Then, since \( |xy| \leq p \), \( y \) consists only of 0's, so \( xz = 0^{p+|y|}10^p \) which is not in the language since \( |y| > 0 \), so \( p + |y| + p \neq 2p \).

(c) \( \{c^na^nb^m | n \geq m \} \).

Solution: Given \( p \geq 1 \), choose \( s = c^pa^pb^p \). Then, \( s \) is in the language and \( |s| = 2p + 4 \geq p \). Given \( x, y, z \) with \( s = xyz \), \( |xy| \leq p \) and \( |y| > 0 \), we choose \( i = 0 \). To see that \( xz \) is not in the language, we consider two cases.

Case 1: \( y \) contains at least one \( c \). Then, \( xz \) contains fewer than 4 \( c \)'s so is not in the language.

Case 2: \( y \) does not contain any \( c \). Then, since \( |xy| \leq p \), \( y \) must contain only \( a \)'s, so \( xz = c^pa^{p-|y|}b^p \) which is not in the language, since \( |y| > 0 \), so \( p - |y| \geq p \).

(d) \( \{a^nb^mc^{2m} | n, m \geq 0 \} \).

Solution: Given \( p \geq 1 \), choose \( s = b^pc^{2p} \). [Note that you must choose \( s \) to contain no \( a \)'s since if there are any \( a \)'s in \( s \), then \( s \) can be pumped by letting \( y = a \).] Then, \( s \) is in the language and \( |s| = 3p \geq p \). Given \( x, y, z \) with \( s = xyz \), \( |xy| \leq p \) and \( |y| > 0 \), we choose \( i = 2 \). Since \( |xy| \leq p \), \( y \) must consist only of \( b \)'s, so \( xz^2 = b^{p+|y|}c^{2p} \) which is not in the language since \( |y| > 0 \), so \( 2(p + |y|) \neq 2p \).