1. Use the Pumping Lemma to show that the following languages are not regular:

(a) \( \{0^{2n}10^n | n \geq 0\} \):

**Solution:** Given \( p \geq 1 \), choose \( s = 0^{2p}10^p \). Then, \( s \) is in the language and \( |s| = 3p+1 \geq p \). Given \( x, y, z \) with \( s = xyz \), \( |xy| \leq p \) and \( |y| > 0 \), we must have \( y = 0^k \) for some \( k \) with \( 1 \leq k \leq p \). Choose \( i = 2 \). Then \( xy^iz = 0^{2p+k}10^p \). Since \( k > 0 \), \( 2p + k \neq 2p \), so \( xy^iz \) is not in the language.

(b) \( \{0^n1^m | n \geq 2m\} \):

**Solution:** Given \( p \geq 1 \), choose \( s = 0^{2p}1^p \). 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 must have \( y = 0^k \) for some \( k \) with \( 1 \leq k \leq p \). Choose \( i = 0 \). Then \( xy^iz = 0^{2p-k}1^p \). Since \( k > 0 \), \( 2p - k \neq 2p \), so \( xy^iz \) is not in the language.

(c) \( \{w\#u | w, u \in \{0, 1\}^* \text{ and } |w| < |u|\} \):

**Solution:** Given \( p \geq 1 \), choose \( s = 0^p\#0^{p+1} \). Then, \( s \) is in the language and \( |s| = 2p + 2 \leq p \). Given \( x, y, z \) with \( s = xyz \), \( |xy| \leq p \) and \( |y| > 0 \), we must have \( y = 0^k \) for some \( k \) with \( 1 \leq k \leq p \). Choose \( i = 2 \). Then \( xy^iz = 0^{p+k}\#0^{p+1} \). Since \( k > 0 \), \( p + k \neq p + 1 \), so \( xy^iz \) is not in the language.

(d) \( \{x_1\#x_2\#x_3 | x_1, x_2, x_3 \in \{a, b\}^* \text{ and } x_2 = x_3^R\} \).

**Solution:** Given \( p \geq 1 \), choose \( s = \#a^p\#a^p \). Then, \( s \) is in the language and \( |s| = 2p + 2 \leq p \). Given \( x, y, z \) with \( s = xyz \), \( |xy| \leq p \), \( |y| > 0 \), choose \( i = 2 \). We show that \( xy^iz \) is not in the language by considering two cases:

- **Case 1:** \( y \) contains \#. Then \( xyz \) contains three \#'s, so is not in the language.

- **Case 2:** \( y \) does not contain \#. Then, since \( |xy| \leq p \), we must have \( y = a^k \) for some \( k \) with \( 1 \leq k \leq p \) and \( xy^iz \) is \( \#a^{p+k}\#a^p \) which is not in the language.

2. Let \( A \) be the language consisting of those strings \( w \) in \( \{0, 1, \#\}^* \) such that either \( w \) starts with 0 or \( w = u\#u \) for some \( u \in \{0, 1\}^* \). \( A \) is not regular. In a Pumping Lemma proof of this, you are given \( p \) and you choose \( s \). For each of the following possible choices of \( s \), state whether or not the choice is a good one. If the choice is bad, provide the decomposition that allows the string to be pumped.

(a) \( s = 0^p1^p \#0^p1^p \)

**Solution:** This is bad choice when \( p \geq 2 \) because if we take \( x = \varepsilon \), \( y = 0 \), \( z = 0^{p-1}1^p \#0^p1^p \), then \( xy^iz \) starts with a 0 for all values of \( i \), so is in the language.
Solution: This is a good choice. Given \(x, y, z\) as in the Pumping Lemma, we choose \(i = 2\). Then, \(xy^iz\) starts with a 1, and the part to the left of the \('#' \) is different from the part to the right, so the string is not in the language.

(c) \(s = (10)^p\#(10)^p\)

Solution: This is a good choice for the same reason as in Part (b).

3. Read the discussion of minimum pumping length given in Problem 1.55 of the text (third US edition) and then give the minimum pumping length for the following languages. Justify your answers.

(a) \(0^+21^+\)

Solution: The minimum pumping length is 4. To see this, first note that \(p = 3\) is not a pumping length because 021 is in the language and it cannot be pumped to stay in the language because no matter what \(y\) is, when you pump down with \(i = 0\), the resulting string \(xz\) is too short to be in the language.

Now suppose that \(s\) is in the language and has length \(\geq 4\). Then, \(s\) has the form \(s = 0^n21^m\) with \(n, m > 0\). If \(n > 1\), we can let \(x = \varepsilon, y = 0\) and \(z = 0^n−121^m\). Then \(|y| = 1, |xy| = 1 \leq 4\) and \(xy^iz\) is in the language for all \(i \geq 0\).

If \(n = 1\), then we must have \(m > 1\) since \(|s| \geq 4\), so we can let \(x = 02, y = 1\) and \(z = 1^{m−1}\). Then, \(|y| = 1, |xy| = 3 \leq 4\) and \(xy^iz\) is in the language for all \(i \geq 0\). Thus, 4 is a pumping length for this language.

(b) \(\{w \in \{0, 1\}^* | w \text{ starts and ends with the same symbol}\}\)

Solution: The minimum pumping length is 2. To see this, we first note that 1 is not a pumping length. This is because \(s = 0\) is a string in the language with length 1, but if \(s = xyz\) with \(|y| > 0\), we must have \(y = 0\) and then \(xz = \varepsilon\) which is not in the language.

To see that 2 is a pumping length, let \(s\) be in the language with \(|s| \geq 2\). If \(|s| = 2\), then \(s\) has the form \(dd\) where \(d\) is either 0 or 1 and we let \(x = \varepsilon, y = d\) and \(z = d\). We have \(|y| = 1, |xy| = 1 \leq 2\), and \(xy^iz\) is in the language for all \(i \geq 0\).

If \(|s| > 2\), then we have \(s = dad\), where \(d\) is either 0 or 1 and \(u\) is a nonempty binary string. We let \(x = d, y\) be the first symbol of \(u\) and \(z\) be the last \(|s|−2\) symbols of \(s\). Then, we have \(|y| = 1, |xy| = 2\) and \(xy^iz\) is in the language for all \(i \geq 0\). Thus, 2 is a pumping length for the language.

(c) \(\{01, 011\}\)

Solution: The minimum pumping length is 4. This is because 3 is not a pumping length since 011 is in the language and cannot be pumped to stay in the language, while there are no strings of length 4 or more in the language, so every string of length 4 or more that is in the language can be pumped.
4. Problem 1.54

(a) To see that \( F \) is not regular, by Problem 1.31, it suffices to show that \( F^R \) is not regular. We do this by showing that \( F^R \) does not have the Pumping Property. Given \( p \), choose \( s = c^p b^p a \). Then \( s \in F^R \) and \( |s| \geq p \). Given \( x, y, z \) as in the Pumping Lemma, we must have \( y = c^k \) for some \( k > 0 \). Choose \( i = 2 \). Then, \( xy^i z = c^{p+k} b^k a \), which is not in \( F^R \).

(b) Choose \( p = 2 \). We show that this is a pumping length for \( F \). Given \( s \in F \) with \( |s| \geq 2 \), we must have \( s = a^i b^j c^k \). We consider cases depending on what \( i, j, k \) are.

Case 1: \( i = 0, j = 0 \). Then we must have \( k > 0 \). Take \( x = \varepsilon, y = c, z = c^{k-1} \). Then \( |xy| \leq p, |y| > 0 \) and for all \( r \geq 0 \), \( xy^r z = c^{k+r-1} \), which is in \( F \).

Case 2: \( i = 0, j > 0 \). Take \( x = \varepsilon, y = b, z = b^{i-1} c^k \). Then \( |xy| \leq p, |y| > 0 \) and for all \( r \geq 0 \), \( xy^r z = b^{i+r-1} c^k \), which is in \( F \).

Case 3: \( i = 1 \). Then we must have \( j = k \). Take \( x = \varepsilon, y = a, z = b^j c^j \). Then \( |xy| \leq p, |y| > 0 \) and for all \( r \geq 0 \), \( xy^r z = a^{i+r-1} b^j c^j \), which is in \( F \).

Case 4: \( i = 2 \). Take \( x = \varepsilon, y = a^a, z = b^j c^k \). Then \( |xy| \leq p, |y| > 0 \) and for all \( r \geq 0 \), \( xy^r z = a^{2r} b^j c^k \), which is in \( F \) (since \( 2r \neq 1 \)).

Case 5: \( i > 2 \). Take \( x = \varepsilon, y = a, z = a^{i-1} b^j c^k \). Then \( |xy| \leq p, |y| > 0 \) and for all \( r \geq 0 \), \( xy^r z = a^{i+r-1} b^j c^k \), which is in \( F \) (since \( i + r - 1 \neq 1 \)).

(Note that the special treatment in Case 4 is necessary because if \( s = aab^j c^k \) we can’t take \( y = a \) since then \( xz = ab^j c^k \) is not in \( F \) if \( j \neq k \).)

(c) The Pumping Lemma states that every regular language has the Pumping Property. It does not state that every language with the Pumping Property is regular, so the result in this problem does not contradict the Pumping Lemma.

5. Give context-free grammars for the following languages:

(a) \( \{0^n 1^n | n \geq 0 \} \):

\[
S \rightarrow 00S0|1
\]

(b) \( \{0^n 1^n | n \geq 2m \} \):

\[
S \rightarrow 00S1|0S|\varepsilon
\]

(c) \( \{x_1 \# x_2 \# x_3 | x_1, x_2, x_3 \in \{a, b\}^* \text{ and } x_2 = x_3^R \} \).

\[
S \rightarrow T\#U
T \rightarrow aT|bT|\varepsilon
U \rightarrow aUa|bUb|\#
\]
6. In class, we gave the following two grammars for the language

\[ L = \{ w \mid w \text{ contains the same number of } a\text{'s and } b\text{'s} \} \]

\[
S \rightarrow \varepsilon | aB | bA \\
A \rightarrow aS | bAA \\
B \rightarrow bS | aBB
\]

and

\[
S \rightarrow aSb | bSa | SS | \varepsilon
\]

For each of these grammars, give both a leftmost derivation and a parse tree for the string \textit{bbabaaba}.

\textbf{Solution:}

A leftmost derivation in the first grammar is:

\[
S \Rightarrow bA \Rightarrow bbAA \Rightarrow bbaSA \Rightarrow bba.A \Rightarrow bbabAA \Rightarrow bbabaSA \\
\Rightarrow bbabaA \Rightarrow bbabaAS \Rightarrow bbabaabA \Rightarrow bbabaabaS \Rightarrow bbabaaba
\]

A leftmost derivation in the second grammar is:

\[
S \Rightarrow bSa \Rightarrow bSSa \Rightarrow bSaSsa \Rightarrow bSaSsa \Rightarrow bSaSSsa \Rightarrow bbabSaSa \\
\Rightarrow bbabaSa \Rightarrow bbabaSaSba \Rightarrow bbabaaba
\]

Parse trees corresponding to these derivations are:
7. (a) Give a context-free grammar that generates the language

\[ A = \{ w \in \{a, b\}^* | w \text{ has at least as many } a\text{'s as } b\text{'s} \} \]
[Hint: In class, we gave two grammars that generate the set of strings with the same number of a’s as b’s. You can use either one of these grammars as part of the grammar you give.]

**Solution:**

\[
S \rightarrow T|TaS \\
T \rightarrow aTb|bTa|TT|\varepsilon
\]

(b) Give a justification that your grammar is correct. Use the same type of argument as we gave in class that the grammars that generate strings with the same number of a’s and b’s are correct. You do not need to reprove the correctness of these grammars.

**Solution:** We know from class that the variable $T$ derives exactly the strings of terminals with the same number of a’s as b’s.

Suppose now that a string $w$ of terminals has at least as many a’s as b’s. We need to analyze the string and see that it matches one of the two rules in the grammar with $S$ on the left side.

If $w$ has the same number of a’s as b’s, then $w$ can be derived from the variable $T$, so the rule $S \rightarrow T$ covers this case.

If $w$ has more a’s than b’s, then we consider a counter that counts each a as +1 and each b as −1. At the end of the string $w$, the count is at least 1, so we can divide up $w$ as $ucv$ where $u,v \in \{a,b\}^*$ and $c \in \{a,b\}$ and the count first reaches +1 after $c$ is read. If $c = b$, then the count would be 2 at the end of $u$, so the count would have to be +1 somewhere in the middle of $u$, contradicting the assumption that the count first reaches +1 when $c$ is read. Thus, $c = a$, so $w = uav$, and the count is 0 at the end of $u$. This means that $u$ has the same number of a’s as b’s, and hence it can be derived from $T$. At the start of $v$, the count is 1 and at the end of $v$, the count is at least 1, so $v$ has at least as many a’s as b’s. Thus the rule $S \rightarrow TaS$ covers this case.

(A full proof would also require showing by induction on the length of the derivation that if a string of terminals $w$ is derivable from $S$, then $w$ has at least as many a’s as b’s, but I was not asking for this.)