Give context-free grammars for each of the following languages.

1. All palindromes in $\Sigma^*$

   **Solution:**
   
   $S \rightarrow \epsilon | 0 | 1 | 0S0 | 1S1$
   
   This is just a recursive definition of “palindrome”.

2. All palindromes in $\Sigma^*$ that contain an even number of 1s

   **Solution:**
   
   $S \rightarrow \epsilon | 0 | 0S0 | 1S1$

3. All palindromes in $\Sigma^*$ that end with 1

   **Solution:**
   
   
   $S \rightarrow 1 | 1A1$
   
   Palindromes that start and end with 1
   
   $A \rightarrow \epsilon | 0 | 1 | 0A0 | 1A1$
   
   All palindromes

4. All palindromes in $\Sigma^*$ whose length is divisible by 3

   **Solution:** Case analysis for the win!

   $S \rightarrow 0A0 | 1A1 | \epsilon$
   
   palindromes, length mod 3 = 0
   
   $A \rightarrow 0B0 | 1B1 | 0 | 1$
   
   palindromes, length mod 3 = 1
   
   $B \rightarrow 0S0 | 1S1$
   
   palindromes, length mod 3 = 2

   **Solution:** Brute force for the win!

   
   $S \rightarrow \epsilon | 000 | 010 | 101 | 111$
   
   $| 000S00 | 001S100 | 010S010 | 011S110$
   
   $| 100S001 | 101S101 | 110S011 | 111S111$

5. All palindromes in $\Sigma^*$ that do not contain the substring 00

   **Solution:**

   $S \rightarrow \epsilon | 1 | 0 | 0A0 | 1S1$
   
   Palindromes with no 00
   
   $A \rightarrow 1 | 1S1$
   
   Palindromes with no 00 that start and end with 1
Harder problems to work on later:

6. \( \{0^{2n}1^n \mid n \geq 0\} \)

**Solution:** \( S \rightarrow \epsilon \mid 00S1 \)

---

7. \( \{0^{m+1}\mid m \neq 2n\} \)  

**Solution:** Intuitively, we can parse any string \( w \in L \) as follows. First, remove the first \( 2k \) \( 0 \)s and the last \( k \) \( 1 \)s, for the largest possible value of \( k \). The remaining string cannot be empty, and it must consist entirely of \( 0 \)s, entirely of \( 1 \)s, or a single \( 0 \) followed by any number of \( 1 \)s.

\[
S \rightarrow 00S1 \mid A \mid B \mid C \quad \{0^{m+1}\mid m \neq 2n\}
\]

\[
A \rightarrow 0 \mid 0A \quad 0^+ \\
B \rightarrow 1 \mid 1B \quad 1^+ \\
C \rightarrow 0 \mid 0B \quad 01^*
\]

---

**Solution:** To simplify notation, let \( \Delta(w) = \#(0,w) - 2\#(1,w) \). Our solution uses the following case analysis. Let \( w \) be an arbitrary string in this language.

- Because \( \Delta(w) \neq 0 \), either \( \Delta(w) > 0 \) or \( \Delta(w) < 0 \).
- If \( \Delta(w) > 0 \), then \( w = 0^iz \) for some integer \( i > 0 \) and some suffix \( z \) with \( \Delta(z) = 0 \).
- If \( \Delta(w) < 0 \), then \( w = x1^j \) for some integer \( j > 0 \) and some prefix \( x \) with either \( \Delta(x) = 0 \) or \( \Delta(x) = 1 \).
- Substrings with \( \Delta = 0 \) are generated by the previous grammar; we need only a small tweak to generate substrings with \( \Delta = 1 \).

We encode this case analysis as a CFG as follows. The nonterminals \( M \) and \( L \) generate all strings where the number of \( 0 \)s is More or Less than twice the number of \( 1 \)s, respectively. The last nonterminal generates strings with \( \Delta = 0 \) or \( \Delta = 1 \).

\[
S \rightarrow M \mid L \quad \{0^{m+1}\mid m \neq 2n\} \quad (\Delta \neq 0)
\]

\[
M \rightarrow 0M \mid 0E \quad \{0^{m+1}\mid m > 2n\} \quad (\Delta > 0)
\]

\[
L \rightarrow L1 \mid E1 \quad \{0^{m+1}\mid m < 2n\} \quad (\Delta < 0)
\]

\[
E \rightarrow \epsilon \mid 0 \mid 00E1 \quad \{0^{m+1}\mid m = 2n \text{ or } 2n+1\}
\]
Solution: Here is another way to encode the logic of the previous solution as a CFG. We either identify a non-empty prefix of 0s or a non-empty prefix of 1s, so that the rest of the string as “balanced” as possible. We also generate strings with $\Delta = 1$ using a separate non-terminal.

| Production | Constraint
|------------|----------------|
| $S \rightarrow AE \mid EB \mid FB$ | \{$0^m 1^n \mid m \neq 2n\}$
| $A \rightarrow 0 \mid 0A$ | $0^+ = \{0^i \mid i \geq 1\}$
| $B \rightarrow 1 \mid 1B$ | $1^+ = \{1^j \mid j \geq 1\}$
| $E \rightarrow \varepsilon \mid 00E1$ | \{$0^m 1^n \mid m = 2n\}$
| $F \rightarrow 0E$ | \{$0^m 1^n \mid m = 2n + 1\}$

Solution: Here is yet another way to encode the logic of the second solution as a CFG. We separately generate all strings of the form $0^{\text{odd}}1^*$, so that we don’t have to worry about the case $\Delta = 1$ separately.

| Production | Constraint
|------------|----------------|
| $S \rightarrow D \mid M \mid L$ | \{$0^m 1^n \mid m \neq 2n\}$
| $D \rightarrow 0 \mid 00D \mid D1$ | \{$0^m 1^n \mid m \text{ is odd}\}$
| $M \rightarrow 00M \mid 00E$ | \{$0^m 1^n \mid m > 2n \text{ and } m \text{ is even}\}$
| $L \rightarrow L1 \mid E1$ | \{$0^m 1^n \mid m < 2n \text{ and } m \text{ is even}\}$
| $E \rightarrow \varepsilon \mid 00E1$ | \{$0^m 1^n \mid m = 2n\}$

8. \{$0,1\}^* \setminus \{0^{2n}1^n \mid n \geq 0\}$

Solution: This language is the union of the previous language and the complement of $0^*1^*$, which is $(0 + 1)^*10(0 + 1)^*$.

| Production | Constraint
|------------|----------------|
| $S \rightarrow T \mid X$ | \{$0,1\}^* \setminus \{0^{2n}1^n \mid n \geq 0\}$
| $T \rightarrow 00T1 \mid A \mid B \mid C$ | \{$0^m 1^n \mid m \neq 2n\}$
| $A \rightarrow 0 \mid 0A$ | $0^+$
| $B \rightarrow 1 \mid 1B$ | $1^+$
| $C \rightarrow 0 \mid 0B$ | $01^*$
| $X \rightarrow Z10Z$ | $(0 + 1)^*10(0 + 1)^*$
| $Z \rightarrow \varepsilon \mid 0Z \mid 1Z$ | $(0 + 1)^*$
9. \( \{ w \in \{0,1\}^* \mid \#(0,w) = 2 \cdot \#(1,w) \} \) — Binary strings where the number of 0s is exactly twice the number of 1s.

Solution: \( S \rightarrow \varepsilon \mid SS \mid 00S1 \mid 1S00 \mid 0S1S0. \)

Let \( L \) denote the language generated by this grammar. For any string \( w \), let \( \Delta(w) = \#(0,w) - 2 \cdot \#(1,w) \). We claim that \( L \) contains every binary string \( w \) such that \( \Delta(w) = 0 \).

Let \( w \) be an arbitrary binary string such that \( \Delta(w) = 0 \). Assume that \( L \) contains every string \( x \) shorter than \( w \) such that \( \Delta(x) = 0 \). There are five cases to consider.

- If \( w = \varepsilon \), the grammar immediately implies \( w \in L \).
- Suppose \( \Delta(x) = 0 \) for some non-empty proper prefix \( x \) of \( w \). Then we can write \( w = xy \), where \( \Delta(y) = \Delta(w) - \Delta(x) = 0 \). The induction hypothesis implies that \( x \in L \) and \( y \in L \). It follows that \( w = xy \in L \).
- Suppose \( \Delta(x) > 0 \) for every non-empty proper prefix \( x \) of \( w \). In this case, \( w \) must start with \( 00 \) and end with \( 1 \). Thus, \( w = 00x1 \) for some string \( x \). We easily observe that \( \Delta(x) = 0 \). So the inductive hypothesis implies \( x \in L \). It follows that \( w = 00x1 \in L \).
- Suppose \( \Delta(x) < 0 \) for every non-empty proper prefix \( x \) of \( w \). In this case, \( w \) must start with \( 1 \) and end with \( 00 \). Let \( 1x \) be the shortest non-empty prefix with \( \Delta(1x) = 1 \). Then \( \Delta(x) = 0 \), and therefore \( x \in L \) by the inductive hypothesis. It follows that \( w = 1x00 \in L \).
- Finally, suppose \( w \) starts with \( 0 \) but \( \Delta(x) < 0 \) for some proper prefix \( x \). Let \( x \) be the shortest non-empty proper prefix of \( w \) with \( \Delta(x) < 0 \). Then \( x = 0y1 \) for some substring \( y \) with \( \Delta(y) = 0 \). Thus, we can write \( w = 0y1z \), and we easily observe that \( \Delta(z) = 0 \). The induction hypothesis implies that \( y \in L \) and \( z \in L \). It follows that \( w = 0y1z0 \in L \).
10. \( \{0,1\}^* \setminus \{ww \mid w \in \{0,1\}^*\} \).

**Solution:** All strings of odd length are in \( L \).

Let \( w \) be any even-length string in \( L \), and let \( m = |w|/2 \). For some index \( i \leq m \), we have \( w_i \neq w_{m+i} \). Thus, \( w \) can be written as either \( x1y0z \) or \( x0y1z \) for some substrings \( x, y, z \) such that \( |x| = i - 1 \), \( |y| = m - 1 \), and \( |z| = m - i \). We can further decompose \( y \) into a prefix of length \( i - 1 \) and a suffix of length \( m - i \). So we can write any even-length string \( w \in L \) as either \( x1x'z'0z \) or \( x0x'z'1z \), for some strings \( x, x', z, z' \) with \( |x| = |x'| = i - 1 \) and \( |z| = |z'| = m - i \).

Said more simply, we can divide \( w \) into two odd-length strings, one with a 0 at its center, and the other with a 1 at its center.

\[
\begin{align*}
S & \rightarrow AB \mid BA \mid A \mid B & \text{strings not of the form } ww \\
A & \rightarrow 0 \mid \Sigma \Sigma & \text{odd-length strings with 0 at center} \\
B & \rightarrow 1 \mid \Sigma \Sigma & \text{odd-length strings with 1 at center} \\
\Sigma & \rightarrow 0 \mid 1 & \text{single character}
\end{align*}
\]