Give regular expressions for each of the following languages over the alphabet \(\{0, 1\}\).

1. All strings containing the substring \(000\).

   Solution: \((0+1)^*000(0+1)^*\)

2. All strings not containing the substring \(000\).

   Solution: \((1+01+001)^*(\varepsilon + 0 + 00)\)

   Solution: \((\varepsilon + 0 + 00)(1(\varepsilon + 0 + 00))^*\)

3. All strings in which every run of \(0\)s has length at least 3.

   Solution: \((1 + 0000^*)^*\)

   Solution: \((\varepsilon + 1)((\varepsilon + 0000^*)1)^*(\varepsilon + 0000^*)\)

4. All strings in which all the \(1\)s appear before any substring \(000\).

   Solution: \((1 + 01 + 001)^*0^*\)

5. All strings containing at least three \(0\)s.

   Solution: \((0 + 1)^*\varepsilon(0 + 1)^*\varepsilon(0 + 1)^*\varepsilon(0 + 1)^*\)

   Solution (clever): \(1^*01^*01^*0(0 + 1)^*\) or \((0 + 1)^*01^*01^*01^*\)

6. Every string except \(000\). [Hint: Don't try to be clever.]

   Solution: Every string \(w \neq 000\) satisfies one of three conditions: Either \(|w| < 3\), or \(|w| = 3\) and \(w \neq 000\), or \(|w| > 3\). The first two cases include only a finite number of strings, so we just list them explicitly, each case on one line. The expression on the last line includes all strings of length at least 4.

   \[\varepsilon + 0 + 1 + 00 + 01 + 10 + 11\]
   \[+ 001 + 010 + 011 + 100 + 101 + 110 + 111\]
   \[+(1 + 0)(1 + 0)(1 + 0)(1 + 0)(1 + 0)^*\]

   Solution (clever): \(\varepsilon + 0 + 00 + (1 + 01 + 001 + 000(1 + 0))(1 + 0)^*\)
7. All strings $w$ such that in every prefix of $w$, the numbers of 0s and 1s differ by at most 1.

**Solution:** Equivalently, strings in which every even-length prefix has the same number of 0s and 1s — $(01 + 10)\epsilon(0 + 1 + \varepsilon)$

8. All strings containing at least two 0s and at least one 1.

**Solution:** There are three possibilities for how the three required symbols are ordered:

- Contains a 1 before two 0s: $(0 + 1)^* 1 (0 + 1)^* 0 (0 + 1)^*$
- Contains a 1 between two 0s: $(0 + 1)^* 0 (0 + 1)^* 1 (0 + 1)^* 0 (0 + 1)^*$
- Contains a 1 after two 0s: $(0 + 1)^* 0 (0 + 1)^* 0 (0 + 1)^* 1 (0 + 1)^*$

So putting these cases together, we get the following:

$$
(0 + 1)^* 1 (0 + 1)^* 0 (0 + 1)^* \\
+ (0 + 1)^* 0 (0 + 1)^* 1 (0 + 1)^* 0 (0 + 1)^* \\
+ (0 + 1)^* 0 (0 + 1)^* 0 (0 + 1)^* 1 (0 + 1)^*
$$

**Solution:** There are three possibilities for how such a string can begin:

- Start with 00, then any number of 0s, then 1, then anything.
- Start with 01, then any number of 1s, then 0, then anything.
- Start with 1, then a substring with exactly two 0s, then anything.

All together: $000^* 1 (0 + 1)^* + 011^* 0 (0 + 1)^* + 11^* 01^* 0 (0 + 1)^*$

Or equivalently: $(000^* 1 + 011^* 0 + 11^* 01^* 0)(0 + 1)^*$

**Solution (clever):** $(0 + 1)^* (101^* 0 + 011^* 0 + 01^* 01)(0 + 1)^*$

9. All strings $w$ such that in every prefix of $w$, the number of 0s and 1s differ by at most 2.

**Solution:** $(0 (01)^* 1 + 1 (10)^* 0)^* (0 + 1 + \varepsilon)$
10. All strings in which the substring \texttt{000} appears an even number of times.
(For example, \texttt{001000} and \texttt{0000} are in this language, but \texttt{00000} is not.)

**Solution:** Every string in \(\{0, 1\}^*\) alternates between (possibly empty) blocks of \texttt{0}s and individual \texttt{1}s; that is, \(\{0, 1\}^* = (0^*1)^*0^*\). Trivially, every \texttt{000} substring is contained in some block of \texttt{0}s. Our strategy is to consider which blocks of \texttt{0}s contain an even or odd number of \texttt{000} substrings.

- Let \(X\) denote the set of all strings in \(\texttt{0}^*\) with an **even** number of \texttt{000} substrings. In particular, we have \(\epsilon \in X\). We easily observe that \(X = \{0^n \mid n = 1 \text{ or } n \text{ is even}\}\) and thus

\[
X = \texttt{0} + (\texttt{00})^*
\]

- Let \(Y\) denote the set of all strings in \(\texttt{0}^*\) with an **odd** number of \texttt{000} substrings. We easily observe that \(Y = \{0^n \mid n > 1 \text{ and } n \text{ is odd}\}\) and thus

\[
Y = \texttt{000}(\texttt{00})^*
\]

- Let \(Z\) denote the set of strings that starts with a run of \texttt{0}s in \(Y\), ends with a different run of \texttt{0}s in \(Y\), and otherwise every run of \texttt{0}s is in \(X\). The set of non-empty runs of \texttt{1}s is \(11^*\), so we immediately have.

\[
Z = Y11^*(X11^*)^*Y
\]

In fact, we can simplify this expression to \(Z = Y1(X1)^*Y\) because \(\epsilon \in X\). Plugging in our earlier expressions for \(X\) and \(Y\) gives us

\[
Z = \texttt{000}(\texttt{00})^*1 \cdot (\texttt{0} + (\texttt{00})^*)^*1 \cdot \texttt{000}(\texttt{00})^*
\]

- Finally, let \(L\) denote the set of all strings in \(\{0, 1\}^*\) with an even number of \texttt{000} substrings.

\[
L = 1^*((X + Z)11^*)(X + Z)1^*
\]

The subexpression \((X + Z)\) matches all maximal substrings that start with \(\texttt{0}\), end with \(\texttt{0}\), and have an even number of \texttt{000} substrings. Any string in \(L\) can be broken into an alternating sequence of runs of \texttt{1}s and strings in \((X + Z)\). In fact, we can simplify this expression to \(L = ((X + Z)1)^*(X + Z)\) because \(\epsilon \in X\). Plugging in our earlier expressions for \(X\) and \(Z\) gives us a complete regular expression for \(L\):

\[
L = ((\texttt{0} + (\texttt{00})^*) + \texttt{000}(\texttt{00})^*1 \cdot ((\texttt{0} + (\texttt{00})^*)^*1 \cdot \texttt{000}(\texttt{00})^*1 \cdot (\texttt{0} + (\texttt{00})^*)^*1 \cdot \texttt{000}(\texttt{00})^*))^*
\]

Whew!