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 0s has length at least 3.
   - Solution: \((1 + 0000^*)^*\)
   - Solution: \((\varepsilon + 1)((\varepsilon + 0000^*)1)(\varepsilon + 0000^*)\)

4. All strings in which 1 does not appear after a substring 000.
   - Solution: \((1 + 01 + 001)^*0^*\)

5. All strings containing at least three 0s.
   - Solution: \((0 + 1)^*0(0 + 1)^*0(0 + 1)^*0(0 + 1)^*\)
   - Solution: \(1^*01^*0(0 + 1)^*\) or \((0 + 1)^*01^*01^*\)

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. The last case includes all strings of length at least 4.

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

- Solution: \(\varepsilon + 0 + 00 + (1 + 01 + 001 + 000(1 + 0))(1 + 0)^*\)

7. All strings \( w \) such that in every prefix of \( w \), the number of 0s and 1s differ by at most 1.
   - Solution: Equivalently, strings that alternate between 0s and 1s: \((01 + 10)^*(\varepsilon + 0 + 1)\)

8. (Hard.) All strings containing at least two 0s and at least one 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: \((000^*1 + 011^*0 + 11^*01^*0)(0 + 1)^*\)
There are three possibilities for how the three required symbols are ordered:

- Contains a 1 before two 0s: 
  \[(0 + 1)^* 1 (0 + 1)^* 0 (0 + 1)^* 0 (0 + 1)^* \]

- 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:

\[(0 + 1)^* 1 (0 + 1)^* 0 (0 + 1)^* 0 (0 + 1)^* \]
\[+ (0 + 1)^* 0 (0 + 1)^* 1 (0 + 1)^* 0 (0 + 1)^* \]
\[+ (0 + 1)^* 0 (0 + 1)^* 0 (0 + 1)^* 1 (0 + 1)^* \]

### Solution:

\[(0 + 1)^* (101^*0 + 000 \, 011^*0 + 01^*01) (0 + 1)^* \]

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

**Solution:** 
\[(0(01)^*1 + 1(10)^*0)^* \cdot (\varepsilon + 0(01)^*(0 + \varepsilon) + 1(10)^*(1 + \varepsilon)) \]

**10** (Really hard.) All strings in which the substring 000 appears an even number of times.
(For example, 0001000 and 0000 are in this language, but 00000 is not.)

**Solution:**

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

Let X denote the set of all strings in 0* with an even number of 000 substrings. We easily observe that \(X = \{0^n \mid n = 1\text{ or } n \text{ is even}\} = 0 + (00)^*\).

Let Y denote the set of all strings in 0* with an odd number of 000 substrings. We easily observe that \(Y = \{0^n \mid n > 1\text{ and } n \text{ is odd}\} = 000(00)^*\).

We immediately have 0* = X + Y and therefore \(\{0, 1\}^* = ((X + Y)1^*)(X + Y)\).

Finally, let L denote the set of all strings in \{0, 1\}^* with an even number of 000 substrings. A string \(w \in \{0, 1\}^*\) is in L if and only if an odd number of blocks of 0s in \(w\) are in Y; the remaining blocks of 0s are all in X.

\[L = ((X1)^*Y1 \cdot (X1)^*Y1)^* (X1)^*X\]

Plugging in the expressions for X and Y gives us the following regular expression for L:

\[\left( ((0 + (00)^*1)^* \cdot 000(00)^*1 \cdot ((0 + (00)^*)1)^* \cdot 000(00)^*1 \cdot (0 + (00)^*)) \right)^* \cdot ((0 + (00)^*1)^* \cdot (0 + (00)^*)) \]

Whew!