Give regular expressions for each of the following languages over the binary 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: Hidden until after the PrairieLearn Guided Problem Set 2 deadline.  

3. All strings in which every run of \( 0 \)s has length at least 3.
   
   Solution: \( (1 + 000^*)^* \)
   
   Solution: \( (\epsilon + 1)((\epsilon + 000^*)1)^*(\epsilon + 000^*) \)  

4. All strings in which every \( 1 \) appears before every substring \( 000 \).
   
   Solution: \( (1 + 01 + 001)^*0^* \)  

5. All strings containing at least three \( 0 \)s.
   
   Solution: \( (0 + 1)^*0(0 + 1)^*0(0 + 1)^*0(0 + 1)^* \)  
   
   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: Hidden until after the PrairieLearn Guided Problem Set 2 deadline.
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:

$$(01 + 10)^* (0 + 1 + \varepsilon)$$

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

**Solution:** Hidden until after the PrairieLearn Guided Problem Set 2 deadline.

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

**Solution:**

$$(0(01)^*1 + 1(10)^*0)^* \cdot (\varepsilon + 0(01)^*(0 + \varepsilon) + 1(10)^*(1 + \varepsilon))$$
10. 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. 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
  \[
  X = 0 + (00)^*
  \]

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

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

In fact, we can simplify this expression to \( Z = Y 1(X 1)^* Y \) because \( \epsilon \in X \). Plugging in our earlier expressions for \( X \) and \( Y \) gives us

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

- Finally, let \( L \) denote the set of all strings in \( \{0, 1\}^* \) with an even number of 000 substrings.
  \[
  L = 1^*((X + Z) 11^*)^*(X + Z) 1^*
  \]

The subexpression \( X + Z \) matches all maximal substrings that start with 0, end with 0, and have an even number of 000 substrings. Any string in \( L \) can be broken into an alternating sequence of runs of 1s 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 \( L \):

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

Whew!