Recall from lecture that a *regular expression* is compact notation for a language (that is, a set of strings). Formally, a regular language is one of the following:

- The symbol $\varnothing$ (representing the empty set)
- Any string (representing the set containing only that string)
- $\boldsymbol{R+S}$ for some regular expressions R and S (representing alternation / union)
- $\boldsymbol{R \cdot S}$ or $\boldsymbol{RS}$ for some regular expressions R and S (representing concatenation)
- $\boldsymbol{R}^*$ for some regular expression R (representing Kleene closure / unbounded repetition)

In the absence of parentheses, Kleene closure has highest precedence, followed by concatenation. For example, $\mathtt{1}+\mathtt{01}^* = \{\mathtt{0}, \mathtt{1}, \mathtt{01}, \mathtt{011}, \mathtt{0111}, \ldots\}$, but $(\mathtt{1}+\mathtt{01})^* = \{\varepsilon, \mathtt{1}, \mathtt{01}, \mathtt{11}, \mathtt{011}, \mathtt{101}, \mathtt{111}, \mathtt{0101}, \ldots\}$.

Give regular expressions for each of the following languages over the binary alphabet $\{\mathtt{0}, \mathtt{1}\}$. (For extra practice, find multiple regular expressions for each language.)

0. All strings.
1. All strings containing the substring `000`.
2. All strings *not* containing the substring `000`.
3. All strings in which every run of `0`s has length at least 3.
4. All strings in which every `1` appears before every substring `000`.
5. All strings containing at least three `0`s.
6. Every string except `000`. *[Hint: Don't try to be clever.]*

More difficult problems to work on later:

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

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

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

10. All strings in which every run has odd length. (For example, `0001` and `100000111` and the empty string ε are in this language, but `000000` and `001000` are not.)

★11. All strings in which the substring `000` appears an even number of times. (For example, `01100` and `000000` and the empty string ε are in this language, but `00000` and `001000` are not.)