1. For each of the following languages over the alphabet $\{0, 1\}$, give a regular expression that describes that language, and *briefly* argue why your regular expression is correct.

(a) All strings except 010.

Solution (brute force):

$$\begin{aligned} &\varepsilon + 0 + 1 + 00 + 01 + 10 + 11 \\ &+ 000 + 001 + 011 + 100 + 101 + 110 + 111 \\ &+ (0+1)(0+1)(0+1)(0+1)(0+1)^* \end{aligned}$$

The first line matches all strings of length that re shorter than 010; the second line matches all strings of length 3 except 010; the last line matches all strings that are longer than 010. ■

Solution (prefix case analysis):

$$\begin{aligned} &\varepsilon + 0 + 01 \\ &+ 010(0+1)(0+1)^* \\ &+ (1 + 00 + 011)(0+1)^* \end{aligned}$$

The first line matches all proper prefixes of 010; the second line matches all strings for which 010 is a proper prefix; the last line matches all strings that have a prefix that is not a prefix of 010. ■

(b) All strings that contain the substring 010.

Solution: $(0+1)^*\,010\,(0+1)^*$ — Anything, then 010, then anything. ■

(c) All strings that contain the subsequence 010.

Solution: $(0+1)^*\,0\,(0+1)^*\,1\,(0+1)^*\,0\,(0+1)^*$

Any string can appear before, after, or inside the subsequence 010. ■

(d) All strings that do not contain the substring 010.

Solution: $1^*(0111^*)^*(1+\varepsilon)$ or $1^*(0+11+111)^*1^*$

Every run of 1s, except possibly at the start and end of the string, has length at least 2. (A *run* of 1s is a maximal nonempty substring consisting entirely of 1s.) Every run of length at least 2 is the concatenation of substrings of length 2 (11) and 3 (111). ■

(e) All strings that do not contain the subsequence 010.

Solution: $1^*0^*1^*$ — Every string in this language has at most one run of 0s. ■

Rubric: 2 points each = 1 for correctness + 1 for explanation. These are not the only correct solutions.

2. Let L be the set of all strings in $\{0,1\}^*$ that contain *at least two* occurrences of the substring `010`.

   (a) Give a regular expression for L, and briefly argue why your expression is correct.

   Solution: $(0+1)^*\,010\,(0+1)^*\,010\,(0+1)^* \;+\; (0+1)^*\,01010\,(0+1)^*$

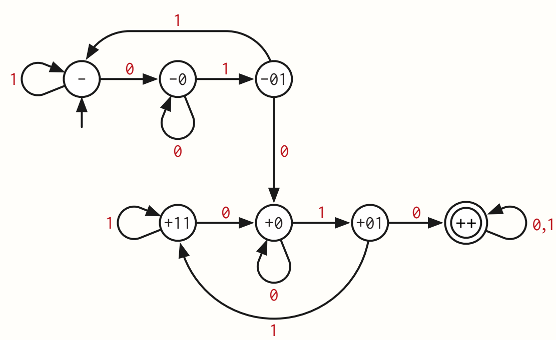
   The first term describes all strings that contain at least two *disjoint* occurrences of the substring `010`. The second term describes all strings that contain the substring `01010`, and therefore contain at least two *overlapping* occurrences of the substring `010`.

   (Neither subexpression attempts the match the *first* two or *last* two occurrences of `010`, and the first subexpression does not attempt to match *adjacent* occurrences of the substring `010`.) ■

   Rubric: 5 points: standard regular expression rubric (scaled). This is not the only correct solution.

   (b) Describe a DFA over the alphabet $\Sigma = \{0,1\}$ that accepts the language L.

   Solution:



The seven states of the DFA have the following meanings:

- `-`: We have not seen the substring `010`, and we have not just read a non-empty prefix of `010`. This is the start state.
- `-0`: We have not seen the substring `010`, and we just read `0`.
- `-01`: We have not seen the substring `010`, and we just read `0` followed by `1`.
- `+11`: We have seen the substring `010` exactly once, and we just read `1` followed by `1`.
- `+0`: We have seen the substring `010` exactly once, and we just read `0`.
- `+01`: We have seen the substring `010` exactly once, and we just read `0` followed by `1`.
- `++`: We have seen the substring `010` at least twice. This is the only accepting state.

■

Rubric: 5 points: standard DFA rubric (scaled). This is not the only correct solution.

3. Let L denote the set of all strings $w \in \{0,1\}^*$ that satisfy *at most two* of the following conditions:

   - The substring 01 appears in w an odd number of times.
   - $\#(1, w)$ is divisible by 3.
   - The binary value of w is *not* a multiple of 7.

   Formally describe a DFA with input alphabet $\Sigma = \{0,1\}$ that accepts the language L, by explicitly describing the states Q, the start state s, the accepting states A, and the transition function δ.

Solution (formal description):

$$Q = \{0,1\} \times \{0,1\} \times \{0,1,2\} \times \{0,1,2,3,4,5,6\}$$
$$s = (1,0,0,0)$$
$$A = \{(a,b,c,d) \mid b = 0 \text{ or } c \neq 0 \text{ or } d = 0\}$$

$$\begin{aligned}
\delta((0,b,c,d),0) &= \big(0,\ b,\ c,\ (2d) \bmod 7\big)\\
\delta((0,b,c,d),1) &= \big(1,\ (b+1) \bmod 2,\ (c+1) \bmod 3,\ (2d+1) \bmod 7\big)\\
\delta((1,b,c,d),0) &= \big(0,\ b,\ c,\ (2d) \bmod 7\big)\\
\delta((1,b,c,d),1) &= \big(1,\ b,\ (c+1) \bmod 3,\ (2d+1) \bmod 7\big)
\end{aligned}$$

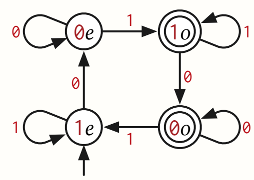
The state (a,b,c,d) indicates the following:

- a is the last symbol read by the DFA, or 1 if the DFA hasn't read anything yet.
- b is the number of times the DFA has read the substring 01, modulo 2.
- c is the number of times the DFA has read the symbol 1, modulo 3.
- d is the binary value of the string red so far, modulo 7.

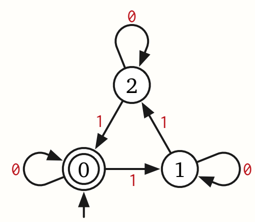
■

Solution (product construction): Our DFA M is the product of three smaller DFAs:

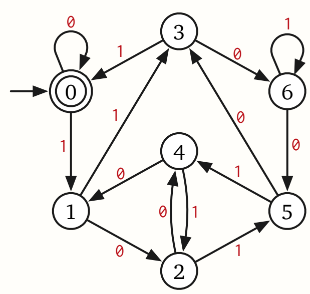
- The first DFA A accepts all strings in which the substring 01 appears an odd number of times. Each state records the last symbol read (of 1 if nothing has been read yet, and whether the DFA has read an even or odd number of 01s.



- The second DFA B accepts all strings where the number of 1s is divisible by 3. Each state records the number of 1s (mod 3) read so far).



- The third DFA C accepts all strings whose binary value is divisible by 3. Each state records the binary value (mod 7) of the string read so far).



Each state of the product DFA M is a triple (a, b, c), where a is a state of A, b is a state of B, and c is a state of C. For example, the start state of M is $(1e, 0, 0)$.

Finally, a state (a, b, c) of M is accepting if and only if $a \in \{0e, 1e\}$ (the number of 01s is even) or $b \neq 0$ (the number of 1s is not divisible by 3) or $c = 0$ (the binary value is divisible by 7). ■

Rubric: 10 points: standard DFA rubric. These are not the only correct solutions.