1. For each of the following regular expressions, (i) give an English description of the set of strings generated (for example, "all strings consisting of 1 or more w's followed by xyz"), and (ii) give an example of a five-character string that can be generated by the expression.

   a. (0|101)*

   b. b(ob)+

   c. 0[0-7]*

   d. (right|wr(ong|ite))

2. Give regular expressions that will generate the following strings.

   a. All strings of lowercase letters in the range a-f where the letters are in ascending lexicographic order.

   b. All strings of a’s and b’s that either start with a and have an even length or start with b and have an odd length.
3. Here is a description from the Java Language Specification (§3.10.1, Integer Literals) for hexadecimal (base 16) integer literals. Write a set of regular expressions that generate hexadecimal literals as described here.

\[
\text{HexIntegerLiteral:}
\]
\[
\text{HexNumeral IntegerTypeSuffixopt}
\]

\[
\text{IntegerTypeSuffix: one of}
\]
\[
I \ L
\]

An integer literal is of type \texttt{long} if it is suffixed with an ASCII letter \texttt{L} or \texttt{l} (ell); otherwise it is of type \texttt{int} (§4.2.1). The suffix \texttt{L} is preferred, because the letter \texttt{l} (ell) is often hard to distinguish from the digit \texttt{1} (one).

A hexadecimal numeral consists of the leading ASCII characters \texttt{0x} or \texttt{0X} followed by one or more ASCII hexadecimal digits and can represent a positive, zero, or negative integer. Hexadecimal digits with values 10 through 15 are represented by the ASCII letters a through f or A through F, respectively; each letter used as a hexadecimal digit may be uppercase or lowercase.

\[
\text{HexNumeral:}
\]
\[
0 \ x \ HexDigits
0 \ X \ HexDigits
\]

\[
\text{HexDigits:}
\]
\[
\text{HexDigit}
\text{HexDigit HexDigits}
\]

The following production from §3.3 is repeated here for clarity:

\[
\text{HexDigit: one of}
\]
\[
0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ a \ b \ c \ d \ e \ f \ A \ B \ C \ D \ E \ F
\]
4. Draw a diagram of a deterministic finite automaton (DFA) that recognizes hexadecimal integer literals as defined by your answer to the preceding question. You can draw this directly; you don’t need to use an algorithm for converting a set of regular expressions to an NFA and then a DFA.