Homework 1
CS 4481

1. (15 points) Construct a DFA accepting each of the following languages:
   (a) $w \in \{a, b\}^*$ such that w starts with *'a'* and contains *'baba'* as a substring
   (b) $w \in \{0, 1\}^*$ such that w contains *'111'* as a substring and does not contain *'00'* as a substring
   (c) $w \in \{a, b\}^*$ such that in w the number of *'a'*s modulo 2 is equal to the number of *'b'*s modulo 3.

2. (6 points) Construct a regular expression to recognize currency in dollars. It is represented as a positive decimal number rounded to the nearest one-hundredth. Such numbers begin with the character \$, followed by a non-zero digit, have commas separating each group of three digits to the left of the decimal point, and end with two digits to the right of the decimal point. For example, \$8,937.43 and \$7,777,777.77. You may use finite closure.

3. (5 points) Give a regular expression for base-16 nonnegative integers (i.e., hexadecimal). Use upper-case letters. The number may not begin with a 0 (i.e., leading zeros are left off). The number will thus *not* be prefixed with 0x.

4. (12 points) Write a regular expression for each of the following languages:
   (a) Given an alphabet $\Sigma = \{0, 1\}$, L is the set of all strings of alternating pairs of 0s and pairs of 1s.
   (b) Given an alphabet $\Sigma = \{0, 1\}$, L is the set of all strings of 0s and 1s that contain an even number of 0s or an even number of 1s.
   (c) Given an alphabet $\Sigma = \{a, b, c, d\}$, L is the set of all strings in which the letters appear in strictly ascending lexicographical order. Strictly ascending means that the same letter can't appear twice.

5. (12 points) Write a regular expression to describe each of the following programming language constructs:
   (a) Any sequence of tabs and blanks (sometimes called *white space*). Use `\t` for tab, and require at least one space or tab.
   (b) Comments in the C programming language. Recall that the only kind of comment in C `/* looks like this */`.
   (c) String constants (without escape characters). This includes all strings that begin with double-quotes, end with (a separate) double-quotes, and do not otherwise contain double-quotes. An example would be to match "abcd" (with double-quotes).