Homework 3

1. Write regular expressions to capture the following regular languages:
   (a) The set of 0-1 strings which have a 1 in every odd position. (Note: even positions may be either 0 or 1.)
   (b) Strings in C. These are delimited at the front and back by double quotes (") and may not contain newline characters. They may contain double quotes or backslash characters if and only if those characters are "escaped" by a preceding backslash. (You may find it helpful to introduce shorthand notation to represent any character that is *not* a member of a small specified set, just to make the picture more readable. For example, if wanting any character that is not a digit between 0 and 9, you could use the shorthand $\neg[0-9]$.)

2. Write a DFA or NFA to recognize the languages described in each part of problem 1.

3. Show the NFA that results from applying the standard construction we saw in class (or you can find in the book in Figure 2.7) to the regular expression $letter(letter|digit)*$. Convert this NFA to a DFA (see Example 2.14 in the book).

4. Give context-free grammars generating the following languages:
   (a) $L_1 = \{a^n b^p \mid 0 \leq p < n\}$.
   (b) $L_2 = \{\ a^n b^n c^m d^m \mid n, m \in \mathbb{N}\ \}$

5. Convert your two grammars from the previous problem to Chomsky Normal Form using the conversion algorithm given in class.

6. Give a (relatively simple) LL(1) grammar for the language which consists of all strings of properly balanced parenthesis and brackets. Use your grammar to construct a parse tree for the string ([ ] ([ ])) [ ] (( )).

7. Consider the following grammar (where non-terminals are in italics):

$$\begin{array}{rcl}
\mathit{stmt} & \rightarrow & \mathit{assignment} \\
 & \rightarrow & \mathit{subroutine\_call} \\
\mathit{assignment} & \rightarrow & \mathrm{id} := \mathit{expr} \\
\mathit{subroutine\_call} & \rightarrow & \mathrm{id}\ (\ \mathit{arg\_list}\ ) \\
\mathit{expr} & \rightarrow & \mathit{primary}\ \mathit{expr\_tail} \\
\mathit{expr\_tail} & \rightarrow & \mathit{op}\ \mathit{expr} \\
 & \rightarrow & \epsilon \\
\mathit{primary} & \rightarrow & \mathrm{id} \\
 & \rightarrow & \mathit{subroutine\_call} \\
 & \rightarrow & (\ \mathit{expr}\ ) \\
\mathit{op} & \rightarrow & + \mid - \mid * \mid / \\
\mathit{arg\_list} & \rightarrow & \mathit{expr}\ \mathit{arg\_tail} \\
\mathit{arg\_tail} & \rightarrow & ,\ \mathit{arg\_list} \\
 & \rightarrow & \epsilon
\end{array}$$

(a) Construct a parse tree for the input `foo(a,b)`.

(b) Give a canonical (rightmost) derivation of this same string.

(c) Prove the grammar is not LL(1).

(d) Describe what you need to do to make the grammar LL(1). (Note: you don't necessarily have to give me the final grammar that results, but describe clearly what you'd have to fix and how you would go about doing that.)