Questions reproduced from textbook: *Programming Language Pragmatics, 4th Edition, Michael Scott, Morgan Kaufman, 2016*. Chapter 2.

Study Homework 1: regular expressions, bottom up parsing.

2.1 Write regular expressions to capture the following.

(d) Floating-point constants in Ada.These match the definition of *real* in Example 2.3, except that (1) a digit is required on both sides of the decimal point, (2) an underscore is permitted between digits, and (3) an alternative numeric base may be specified by surrounding the nonexponent part of the number with pound signs, preceded by a base in decimal (e.g., 16#6.a7#e+2). In this latter case, the letters a . . f (both upper- and lowercase) are permitted as dig- its. Use of these letters in an inappropriate (e.g., decimal) number is an error, but need not be caught by the scanner.

(f) Financial quantities in American notation. These have a leading dollar sign ($), an optional string of asterisks (*—used on checks to discourage fraud), a string of decimal digits, and an optional fractional part consisting of a dec- imal point (.) and two decimal digits. The string of digits to the left of the decimal point may consist of a single zero (0). Otherwise it must not start with a zero. If there are more than three digits to the left of the decimal point, groups of three (counting from the right) must be separated by commas (,). Example: $**2,345.67. (Feel free to use "productions" to define abbreviations, so long as the language remains regular.)

2.14 Consider the language consisting of all strings of properly balanced parentheses and brackets.
(a) Give LL(1) and SLR(1) grammar for this language.

(b) Give the corresponding LL(1) and SLR(1) parsing tables.

(c) For each grammar, show the parse tree for ([]([]))[](()).

(d) Give a trace of the actions of the parsers in constructing these trees.

2.18 Consider the following LL(1) grammar for a simplified subset of Lisp:

$P \rightarrow E$ $$

$E \rightarrow$ atom | ' E | (E Es)

$Es \rightarrow E\ Es$ | ε

(a) What is FIRST(Es)? FOLLOW(E)? PREDICT($Es \rightarrow \varepsilon$)?
(b) Give a parse tree for the string (cdr '(a b c)) $$.
(c) Show the left-most derivation of (cdr '(a b c)) $$.
(d) Show a trace in the style of Fig 2.21 of a table-driven top-down parse of this same input.

If you want to look at harder questions, try these

2.9 (a) Describe in English the language defined by the regular expression a**(** b a* b a* **)** * . Your description should be a high-level characterization—one that would still make sense if we were using a different regular expression for the same language.

(b) Write an unambiguous context-free grammar that generates the same language.

(c) Using your grammar from part (b), give a canonical (right-most) derivation of the string

b a a b a a a b b.

2.15 Consider the following context-free grammar.

$$G \rightarrow G\ B \mid G\ N \mid \varepsilon$$
$$B \rightarrow (\ E\ )$$
$$E \rightarrow E\ (\ E\ ) \mid \varepsilon$$
$$N \rightarrow (\ L\ ]$$

$$L \rightarrow L\ E \mid L\ (\mid \varepsilon$$

(a) Describe, in English, the language generated by this grammar. (Hint: *B* stands for "balanced"; *N* stands for "nonbalanced".) (Your description should be a high-level characterization of the language—one that is indepen- dent of the particular grammar chosen.)

(b) Give a parse tree for the string (( ] ( ) .

(c) Give a canonical (right-most) derivation of this same string.

(d) What is FIRST(*E*) in our grammar? What is FOLLOW(*E*)? (Recall that FIRST and FOLLOW sets are defined for symbols in an arbitrary CFG, regardless of parsing algorithm.)

(e) Given its use of left recursion, our grammar is clearly not LL(1). Does this language have an LL(1) grammar? Explain.