This assignment covers context free grammars and parsing. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work, and you should indicate in your submission who you worked with, if applicable. Assignments can be submitted electronically through Gradescope as a PDF by 11:59 PM PDT. A \LaTeX{} template for writing your solutions is available on the course website.

1. (8 pts) Give the context-free grammar (CFG) for each of the following languages:

   (a) (2 pts) The set of all strings over the alphabet \{int, [, ], ;\} representing nested arrays of integers, e.g. “[[int; int; [int]]; int; [ ]]”. Note that int stands for any integer value and should be a terminal in your CFG. Additionally, each array can contain both integers and other arrays (remember that syntactic analysis is not the place for type checking). Items in the array are separated by semicolons (;).

   (b) (2 pts) The set of all strings over the alphabet \{0, 1\} with at least 2 more 0’s than 1’s.

   (c) (2 pts) The set of all strings over the alphabet \{0, 1\} in the language \(L = \{0^i 1^j 0^k | i \neq j \lor j \neq k\}\).

   (d) (2 pts) The set of all strings over the alphabet \{0, 1, ..., 9, *, +, (, ), ep\} representing valid regular expressions over numbers formed by the digits 0 to 9. In this alphabet ep stands for the \(\epsilon\) symbol in the regular expression, and + is the union operator.

2. (18 pts) Consider the following CFG for expressions made of the terminal symbols \{a, b, <, >, ?, !, @, [, ]\}:

   \[
   E \rightarrow UV \mid EBE \mid V \mid [E] \\
   V \rightarrow a \mid b \\
   U \rightarrow < | > \\
   B \rightarrow ? \mid ! \mid @ \tag{1}
   \]

   (a) (3 pts) Is the grammar as given ambiguous? If yes, give an example of an expression with two parse trees under this grammar. If not, explain why that is the case.

   (b) (3 pts) You are told that, in this grammar, ?, ! and @ are right-associative binary operators and are given in order of ascending precedence (i.e. ? has the lowest precedence, followed by ! and @ has the highest). Unary operators < and > have even higher precedence. Transform the CFG given above by eliminating ambiguity (if needed) and left recursion, as well as performing left factoring. List the resulting grammar. Is this grammar suitable for top-down parsing?

   (c) (3 pts) Give the FIRST and FOLLOW sets for each non-terminal in the grammar you just constructed.

   (d) (3 pts) Construct the LL(1) parsing table for the grammar of the previous question. Is this grammar LL(1)? If the grammar is not LL(1), highlight the conflicts.

   (e) (6 pts) Show the operation of the nonrecursive predictive parser using the previous table on the input string “a@b? > a”. At each step, you must indicate the contents of the stack, the remaining input string, and the action taken by the parser (e.g. the production used as per the parsing table, the matched terminal or the fact that the parser accepted the string, found an error or got stuck).
3. **(9 pts)** Consider the following grammar for if-then-else expressions:

\[
E \rightarrow \text{if } C \text{ then } E \\
\quad \quad | \ MIF \\
MIF \rightarrow \text{if } C \text{ then } MIF \text{ else } E \\
\quad \quad | \ C \\
C \rightarrow \text{id} \ | \ ...
\]

(2)

Is this grammar ambiguous or not? Why?

4. **(20 pts)** The following is a grammar for an abstracted Lisp-like language, in which every expression is represented as a list containing an operator and any number of operands, which can in turn be expressions or identifiers.

\[
P \rightarrow \ ( \text{op} \ C) \\
C \rightarrow ) \mid E \ C \\
E \rightarrow \ ( \text{op} \ C \mid \text{id}
\]

(3)

(a) **(8 pts)** Construct the DFA for viable prefixes of this grammar using LR(0) items. You may either draw the DFA or list its states and its transition table.

(b) **(4 pts)** Identify any shift-reduce and reduce-reduce conflicts in this grammar under SLR(1) rules.

(c) **(8 pts)** Assuming that an SLR(1) parser resolves shift-reduce conflicts by choosing to shift, show the operation of such a parser on the input string \(\text{(op id (op (op id))}\).

5. **(9 pts)** Consider the following CFG, where the words in **bold** are terminals:

\[
S \rightarrow S \ P \mid P \\
P \rightarrow L \ a \ T \mid \text{and another} A \ T \mid A \ T \mid T \ T \ D \\
D \rightarrow \text{duck} \\
A \rightarrow \text{funny} \mid \text{little} \mid \text{fuzzy} \\
L \rightarrow \text{here’s} \mid \text{there’s} \\
T \rightarrow \text{llama}
\]

(4)

(a) **(3 pts)** Is this grammar left-recursive? Why or why not?

(b) **(3 pts)** Is this grammar, as given, LL(1)? Why or why not?

(c) **(3 pts)** Is this grammar, augmented by \(S' \rightarrow S\), SLR(1)? Why or why not? To save you some time and pain on this question and the next, we include (on the next page) a partial diagram of the LR(0) automaton for this grammar, which includes all the states, but omits (only) the transitions to states \(q_{16}\) to \(q_{22}\) for the sake of readability.