Sample Problems for Syntax Analysis

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

   (a) **(2 pts)** The set of nested addition expressions formed using the alphabet $\{int, +, (,), [,]\}$. The associativity of the additions must be established with brackets and adjacent brackets should be of different types. For example, "[(int + int) + int]" and "int + [int + (int + int)]" are strings in the language while "int + int + int" and "(int + (int + int))" are not in the language.
   Answer:

   (b) **(2 pts)** The set of all strings over the alphabet $\{a, b\}$ with more "a"s than "b"s.
   Answer:

   (c) **(2 pts)** The set of all strings over the alphabet $\{a, b, c\}$ in the language $L : \{a^i b^j c^k \mid j = i + k\}$.
   Answer:

   (d) **(2 pts)** The set of all strings over the alphabet $\{0, 1\}$ in the language $L : \{0^i 1^j 0^k \mid i \neq j \vee j \neq k\}$.
   Answer:

2. (a) Left factor the following grammar:

$$P \to x \mid P \Rightarrow P \mid P \Leftarrow P \mid (P) \Leftrightarrow (P) \mid P; P; \tag{1}$$

   Answer:

   (b) Eliminate left recursion from the following grammar:

$$\begin{aligned} A &\to Aa \mid ABA \mid \epsilon \\ B &\to Bb \mid BB \mid \epsilon \end{aligned} \tag{2}$$

   Answer:

3. **(9 pts)** Consider the following grammar:

$$\begin{aligned} A &\to uA \mid wuA \mid B + B \mid \epsilon \\ B &\to bB \mid CB \mid \epsilon \\ C &\to cAw \end{aligned} \tag{3}$$

   A, B, and C are the non-terminals in the grammar.

   (a) Construct the FIRST sets for the grammar.
   Answer:

   (b) Construct the FOLLOW sets for the grammar.
   Answer:

   (c) Construct the LL(1) parse table for the grammar.
   Answer:

(d) Show the sequence of stack and input configurations that occur during an LL(1) parse of the string "$cuw + b$". The stack should contain a single A at the beginning of the parse.
Answer:

4. Suppose you encounter the following grammar G:

$$
\begin{aligned}
S' &\to S \\
S &\to Aa \\
S &\to Bb \\
A &\to Ac \\
A &\to \epsilon \\
B &\to Bc \\
B &\to \epsilon
\end{aligned}
$$

You want to implement G using an SLR(1) parser (note that we have already added the $S' \to S$ production for you).

(a) Why is left recursion preferable in shift-reduce parsing? [Hint: consider left and right recursive grammars for the language a^*. What happens if your input has a million a's?]
Answer:

(b) Construct the first state of the LR(0) machine, compute the FOLLOW sets of A and B, and point out the conflicts that prevent the grammar from being SLR(1).
Answer:

(c) Show that changing productions 4 and 6 to be right-recursive solves the problem in this case (show that the grammar is SLR(1)). Explain the intuition behind this result.
Answer:

5. Consider the following CFG, which has the set of terminals $T = \{\textbf{stmt}, \textbf{\{}, \textbf{\}}, \textbf{;}\}$. This grammar describes the organization of statements in blocks for a fictitious programming language. Blocks can have zero or more statements as well as other nested blocks, separated by semicolons, where the last semicolon is optional. (P is the start symbol here.)

$$
\begin{aligned}
P &\to S \\
S &\to \textbf{stmt} \mid \textbf{\{}B \\
B &\to \textbf{\}} \mid S\textbf{\}} \mid S\textbf{;}B
\end{aligned}
$$

(a) Construct a DFA for viable prefixes of this grammar using LR(0) items.
Answer:

(b) Identify any shift-reduce and reduce-reduce conflicts in this grammar under the SLR(1) rules.
Answer:

(c) 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 **{stmt;stmt}**
Answer: