CS 415: Compilers - Problem Set 1
Due Date: Wednesday, Feb 16, Beginning of Lecture

Problem 1

Construct a nondeterministic finite automaton (NFA) for the regular expression \((ab|c)^*d\) using Algorithm 3.3 (p.122 in Aho, Sethi, Ullman).

Problem 2

Convert your NFA for problem 1 to a DFA using Algorithm 3.2 (p.118 in Aho, Sethi, Ullman).

Problem 3

Here is an attempt to remove the dangling-else ambiguity for if-then-else statements:

1 \( \langle \text{stmt} \rangle \) ::= \( \text{if} \langle \text{expr} \rangle \text{ then} \langle \text{stmt} \rangle \) |
   \( \langle \text{matched_stmt} \rangle \)

2 \( \langle \text{matched_stmt} \rangle \) ::= \( \text{if} \langle \text{expr} \rangle \text{ then} \langle \text{matched_stmt} \rangle \text{ else} \langle \text{stmt} \rangle \) |
   \( \text{other} \)

Show that this grammar is still ambiguous.

Problem 4

Consider the following grammar with start symbol \(S\):

1 \( S ::= \langle L \rangle \) |
2 \( a \)
3 \( L ::= L, S \) |
4 \( S \)

1. Give parse trees for the sentences \(a, a\) and \((a, a, a)\).
2. Construct a leftmost and a rightmost derivation for the sentence \((a, a, a)\).
3. What language does this grammar generate?

Problem 5

1. Eliminate the left-recursion from the grammar in Problem 4.
2. Compute the FIRST and FOLLOW sets for the resulting grammar.
3. Compute the LL(1) parse table for the resulting grammar. Is the grammar LL(1) or not? Justify your answer.

4. If the resulting grammar is LL(1), show the behavior of the LL(1) parser (sequence of states [stack, input, next action]) on sentence \((a, (a, a))\).

5. If the resulting grammar is LL(1), write a recursive descent parser for it. Insert code into the recursive descent parser such that the parser returns the maximum nesting depth of any \(a\) in a sentence of the language. Examples: \(\text{MaxDepth}[ (a, a) ] = 1, \text{MaxDepth}[ (a, (a, a)) ] = \text{MaxDepth}[ (((a))) ] = 3\).