COMS W4115
Programming Languages and Translators
Homework Assignment 2

Prof. Stephen A. Edwards
Columbia University

Due March 3rd, 2009
at 11:59 PM

On-campus students: submit solution on paper; do not email.

CVN students: FAX the solutions to CVN.

Write both your name and your Columbia ID (e.g., se2007) on your solutions.

Do this assignment alone. You may consult the instructor or a TA, but not other students.

1. Scanners

   (a) Using Ocamllex-like syntax, write a scanner for C's floating point numbers, as defined by Ritchie.

   > A floating constant consists of an integer part, a decimal point, a fraction part, an `e`, and an optionally signed integer exponent. The integer and fraction parts both consist of a sequence of digits. Either the integer part or the fraction part (not both) may be missing; either the decimal point or the `e` and the exponent (not both) may be missing.

   Hint: make sure your scanner accepts constants such as `1.` `0.5e-15` `.3e+3` `.2` `1e5` but not integer constants such as `42`

   (b) Draw a DFA for a scanner that recognizes and distinguishes the following set of keywords. Draw accepting states with double lines and label them with the name of the keyword they accept. Follow the definition of a DFA given in class.

   `if else ifelse union unsigned void volatile`

2. Dragon book 2ed, Exercise 3.7.3, p. 166:

   Construct nondeterministic finite automata for the following regular expressions using Algorithm 3.23 (p. 159, shown in class), then use the subset construction algorithm to construct DFAs for them using Algorithm 3.20 (p. 153, also shown in class).

   (a) $(a\,|\,b)^*$

   (b) $(a^*\,|\,b^*)^*$

   (c) $((\varepsilon\,|\,a)b^*)^*$

   I suggest you use text to label the states of the DFA while running the subset construction algorithm. You do not have to use the graphical style in the lecture notes.

3. Using the grammar

$$S \to (L)\,|\,a$$
$$L \to L,S\,|\,S$$

   (a) Construct a rightmost derivation for $(a,(a,a))$ and show the handle of each right-sentinel form.

   (b) Show the steps of a shift-reduce (bottom-up) parser corresponding to this rightmost derivation.

   (c) Show the concrete parse tree that would be constructed during this shift-reduce parse.

4. Build the LR(0) automaton for the following ambiguous grammar (**if** and **else** are terminals; the third rule indicates t may be the empty string.). Show the state in which the shift/reduce conflict appears.

$$s' \to s$$
$$s \to \textbf{if}\ s\ t$$
$$t \to$$
$$t \to \textbf{else}\ s$$

   Check your work by running "ocamlyacc -v" on the grammar below.

```
%token IF ELSE
%start s
%type <int list>s

%%

s : IF s t      { [] }

t : /* empty */ { [] }
  | ELSE s      { [] }
```