COMS W4115
Programming Languages and Translators
Homework Assignment 2

Prof. Stephen A. Edwards
Columbia University

Due June 27th, 2011
at 11:59 PM your time

CVN students: FAX the solutions to CVN.

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.

   `abort abs accept access else elsif for subtype type`

2. 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) $(a \mid ab)^*$

   (b) $(a\,(\epsilon \mid b))^*$

   (c) $a\,(a \mid b)^*\,b$

   Number the NFA states; use the numbers to label DFA states while performing subset construction, e.g., like Figure 3.35 (p. 155).

3. Using the grammar

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

   (a) Construct a rightmost derivation for $(a, (a, a))$ and show the handle of each right-sentential 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**, **else**, and **null** are terminals; the third rule indicates T may be the empty string. Indicate the state in which the shift/reduce conflict appears.

$$S' \to S$$
$$S \to \textbf{if}\ S\ T$$
$$S \to \textbf{null}$$
$$T \to$$
$$T \to \textbf{else}\ S$$

   Check your work by running "ocamlyacc -v" on the grammar below and looking through the ".output" file.

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

%%

s : IF s t       { 0 }
  | NULL         { 0 }

t : /* empty */ { 0 }
  | ELSE s       { 0 }
```