COMS W4115
Programming Languages and Translators
Homework Assignment 2

Prof. Stephen A. Edwards
Columbia University

Due June 29, 2015
at 11:59 PM

CVN students: submit online.

Include your name and your Columbia ID (e.g., se2007).

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

1. Using Ocamllex-like syntax, write a scanner for C's floating point numbers following the definition in K&R 2ed.

   > A floating constant consists of an integer part, a decimal part, a fraction part, an `e` or `E`, an optionally signed integer exponent and an optional type suffix, one of `f`, `F`, `l`, or `L`. 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. The type is determined by the suffix; `F` or `f` makes it `float`, `L` or `l` makes it `long double`, otherwise it is `double`.

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

2. 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 (single) keyword they accept. Follow the definition of a DFA given in class.

   `chan chanin chanout width with if end endif elseif`

3. 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) $(ab \mid b)^*$

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

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

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

4. Using the grammar

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

   (a) Construct a rightmost derivation for $((a,a),(a,a))$ 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.

5. 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.

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

   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 }
```