CS 316 Fall 2009
All projects in this course must be completed **individually and independently**. All programs must be written in Java or C++.

PROJECT 1: Finite Automaton to Recognize Tokens
Due: In the class on 10/13/09

Consider the following BNF defining 22 token categories <int> through <comma>:

<letter> → a | b | ... | z | A | B | ... | Z
<digit> → 0 | 1 | ... | 9
<int> → {<digit>}$^+$
<id> → <letter> { <letter> | <digit> }
<float> → {<digit>}$^+$ "." {<digit>} | "." {<digit>}$^+$
<floatE> → <float> (E|e) [+|−] {<digit>}$^+$
<add> → +
<sub> → −
<mul> → *
 → /
<or> → "||"
<and> → "&&"
<inv> → !
<lt> → "<"
<le> → "<="
<gt> → ">"
<ge> → ">="
<eq> → "=="
<neq> → "!="
<LParen> → "("
<RParen> → ")"
<LBrace> → "{"
<RBrace> → "}"
<comma> → ","

The integer or fractional parts, but not both, of the floating-point numbers may be empty. In this project, you are to draw a **state transition diagram** of a DFA to accept the above 22 token categories. The DFA should have 22 final states corresponding to the 22 token categories. Note that "||", "&&", "<=", ">=", "==", and "!=" consist of two characters and require two transitions. Make sure that your automaton is **deterministic**: at most one transition for each (state, input char) pair and no transition on the empty string ε.

The above token set is used for a small, type-free functional language specifically designed for our projects. Our project plan for the semester is to implement a lexical analyzer for this functional language in Project 2, a top-down parser and an intermediate code generator in Projects 3 and 4.

Submission

A **hardcopy** of the state transition diagram of your DFA is due in the class on 10/13/09. A legibly handwritten diagram is fine.