CMSI 585
PROGRAMMING LANGUAGES (GRADUATE LEVEL)

Fall 2006

Assignment 0919

This assignment seeks to flex your language specification muscles. We will use unit tests to empirically test how well you've specified a language's grammar.

Not for Submission

These items are actually prerequisites for the work to be submitted. The necessary links/addresses can be found on the course Web site.

1. Download and install *javacc*, the Java Compiler Compiler. Both the site and the download include additional documentation and examples that supplement our discussion in class. Prof. Toal also has an introductory tutorial at his site.
2. If you haven't already done so, download and install JUnit 4. If you're using Eclipse 3.2 or higher, then you already have it.
3. If necessary, re-read Chapter 2.

For Submission

A simple language called "Functionator" — actually a subset of many languages that you already know — is given in plain English below:

> The Functionator language captures the syntax that is commonly used for invoking functions with arguments. A Functionator program is a non-empty sequence of assignment statements terminated by semicolons; the assignment operator is ":=" a la Pascal. The left side of the assignment statement is always an identifier, and the right side may be another identifier, an integer literal, or a function call of the form *func_name*(arg_1, arg_2, ... , arg_n). The argument list "arg_1, arg_2, ... , arg_n" is a comma-separated list of identifiers, integer literals, or additional function calls. All expressions (identifiers, literals, functions) are "negatable" and may thus be preceded by the negative sign ("–"). Identifiers for both assigned variables and function names may begin with any Latin letter ('A' to 'Z', 'a' to 'z') or the underscore ("_") character followed by any number of Latin letters, Arabic digits ('0' to '9'), or underscores. Whitespace/skip characters consist of the space, tab, newline, and return characters.
>
> A brief sample:
>
> *alpha := foo(bar, 59, buzz(98, 99, z(12)), –23);*
> *beta := _hidden(500);*
> *x := 22;*
> *y := x;*

As you can see, this is pretty much the function-calling subset that can be found in most languages. Thus, if any details seem to be missing from the above specification, you can use your knowledge of existing languages to make an educated guess. Of course, you can always ask me.

For this language, provide the following:

1. The microsyntax and macrosyntax of the language, with *program* as the start symbol.
2. A *javacc* parser definition of that language. Name the parser *Functionator*, and have the classes go into the package *func.lang*. Name this file "*your-last-name*–Functionator.jj."
3. A JUnit 4 test suite for the parser. The test suite should determine whether your parser properly decides on what strings belong to the language. Make the tests as thorough as you like, and make sure that you also include strings that do *not* belong to the language as well as those that do. Put this test suite in the *func.test* package and call it "*your-last-name*_ParserTest."

Submit (1) on hardcopy, and submit (2) and (3) via CVS, under the *homework/cmsi585/functionator* directory. Put the .jj file directly within this directory, and place your Java files in the appropriate directories within *src/*. For example, your test suite source file would live in *src/func/test* inside *homework/cmsi585/functionator*.

Tag the files in *functionator* with *hw0919*.