Prof. Jingke Li (FAB 120-06, li@cs.pdx.edu), Classes: MW 4:40-6:30pm, NH454, Office Hours: MW 2-3pm.

Project 1: MINI Lexer

(Due Monday 10/10/11)

This project is to implement a lexer for the MINI programming language using the lexer-generator tool, JavaCC. You should first read Section 2 (Lexical Issues) of the *MINI Programming Language (v1.6)* manual to learn its token definitions.

Lexical Errors

For the MINI language, there are five kinds of possible lexical errors:

- *Illegal character* — a character that is not in MINI's vocabulary (e.g. `#`)
- *Integer over limit* — an integer whose value is over the limit $2^{31} - 1$ (i.e. 2147483647)
- *ID over limit* — an ID whose length is over the limit 255
- *String over limit* — a string whose length is over the limit 255 (not counting the quotes)
- *Float over limit* — a floating-point value that is beyond the 32-bit limit (1.4E-45 – 3.4028235E38).

Requirements

- Name your lexer program `miniLexer.jj` and place it in the subdirectory `lexer` in the `proj1` directory (see below). Include the name `lexer` as the package name in the header of your file:

```
PARSER_BEGIN(miniLexer)
package lexer;
public class miniLexer {
}
PARSER_END(miniLexer)
```

  Do not include any driver routine in the program.

- Your lexer needs to detect the first four kinds of lexical errors. Since MINI does not support the exponent form of floating-point literals, there is no practical need to detect the "float-over-limit" error.
- Your lexer only needs to detect and report the first error in a given program — it may simply terminate afterwards. There is no requirement on the exact form of the error reporting, but the error message should contain the nature of the error (i.e. which kind) and the location (i.e. line and column) where it occurred. You may want to take a look at the JavaCC routine `TokenMgrError()`.
- Your lexer's token output should match *exactly* with the ref file (see below). The error message, however, does not have to match exactly.

Testing Your Program

A tar file, `proj1-code.tar`, is provided to you. When untarred, it creates a directory, `proj1`, which contains:

- a `lexer` subdirectory — You'll find a driver program, `Testexer.java` in this directory. You should put your `miniLexer.jj` file here as well. The driver repeatedly calls the lexer to get tokens and then prints them out.

- a `tst` subdirectory — It contains a set of tests for the lexer. Each test consists of an input (e.g. `test01.java`), an expected output (e.g. `test01.tkn.ref`), and an error message log file if there is any error (e.g. `test04.lerr.ref`).
- `Makefile` — To compile your lexer, simple do

  ```
  make lexer
  ```

- `runl` — a shell script for running the tests. Usage example:

  ```
  ./runl tst/test*.java
  ```

  When it runs silently, it means that your lexer's output matches the ref output.

CS Machines and JavaCC

Your program should be compiled and tested on either the CS linux system or the CS solaris system. JavaCC is available on both systems. I suggest that you use the linux system, since there are more machines available and they run faster. The names of the linux and solaris servers are `linuxlab.cs.pdx.edu` and `unix.cs.pdx.edu`, respectively. You may remotely login to them. The CS Linux Lab is located in FAB 88-10.

Project Submission

Submit a single file, `miniLexer.jj`, through the "Dropbox" on the D2L class webpage. (*Don't forget to include your name in your program.*).