CS530, Spring 2019, Program Assignment #3

23 Apr 2019

You shall develop, test, and deliver a simple parser that can evaluate statements.

STATEMENT PARSER REQUIREMENTS:
You shall develop a grammar and implement a parser which recognizes valid statements as described below in the assignment specification. You may develop your code using C, C++, or flex & bison; if you develop/test code on a different machine than edoras, be sure to use the same versions of these tools! C/C++ developers, you may use all libraries found in the directories on edoras, the use of other non-standard libraries must be cleared with me first. You shall develop a file named README. The README file **shall contain** (in addition to the information required at the top of all project files):

* **The grammar (use BNF)** describing a valid statement (rules specifying what can be accepted for an expression)
* A **listing** of all files in the project
* Instructions on how to compile and run the program
* Issues, if any (potential errors, implementation issues)
* Lessons learned, significant findings
* Any additional information we should know when grading your assignment

SPECIFICATION:
Your program shall read a file or from standard input, scan the input, and determine if the statement(s) is/are valid. Your program shall print out the statement, and a pass/fail. If it failed, you shall print out why.

Note: id :== identifier, exp :== expression, op :== operator, char :== character

Your compiler will recognize the following as valid statements:

- **assignment**
- **expression**

assignment shall have the form:
id = exp;

expression shall have the form:
id op id {op id} -- any length as long as pairs of op and id are added

A parenthesis pair may be used to group any id op id combination. Therefore:
id op (id op id) op id **AND** id op id op (id op id) -- valid expressions

Note - each id, op, =, and ; will have a space as it's precedent and antecedent. The open parenthesis will have a space as it's precedent but it may or may not have a space immediately following it. The converse is true for the closing parenthesis (guaranteed a space as it's antecedent but not necessarily preceding it).

An **id** shall be made up of any combination of **digits** and **char**. The first position of the identifier must contain a char
A **digit** is one of: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
A **char** is one of: a, b, c, d, e, f, g, h, i, j, k, l m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

An **op** is one of:
+, -, *, /, %

Your program shall read from a file named "**ex.txt**" which contains examples of both good and bad statements

TEAMS:
You shall work on this individually, not teams.

ADDITIONAL REQUIREMENTS:
README file - you shall create a README file; consult the instructions for README file content on the course Blackboard. Also, your source files SHALL CONTAIN sufficient comments for making the source easy to read. Points will be taken off for poorly (or non) commented source or inadequate README file documentation.

Compiler and make (and Makefile) – You shall use gcc/g++ or flex & bison. You shall use make to read a makefile to compile your program for this assignment. Name your executable '**exp**'.

Test files – I will post a testfile and expected output file for you to use during development and test. I do recommend that you prepare your own additional test file(s) and if you do, include those in your turnin.

Make sure that all files (README, source files, header files, Makefile) contains your name and RedID.

TURNING IN YOUR WORK:
The assignment is due at 1730, Wednesday, 8 May 2019

Your project shall source file(s), a Makefile, and a README file. To turn in your project, all files shall be in your class account on edoras in a directory named "a3" (~/a3) and upload a tarball/zipfile which includes all project files to Blackboard for Lab #3, entering any comments in the assignment's turnin.