CSE341 – Programming Languages (Fall 2020)

Assignment #1 – Lexing and Flexing.

Handed out: Wednesday October 28, 2020.

Due: 11:55pm Saturday November 15, 2020

SS

Hand-in Policy:

- Source codes must be uploaded to Moodle with a compressed format. File name must be studentID.zip
- No late submissions will be accepted.
- Your submission must contain a folder for each part of the of assignment. Folders should be named Flex_Lexer and Lisp_Lexer. Each folder should contain the appropriate and necessary code files in order to compile and test your submissions. Whole assignment should be in a folder named **your_student_number**. And this folder should be zipped as **your_student_number_assignment_1.zip.**
- **We only require source files. No output or input file should be placed into the submission.**
- **BE CAREFUL! If your submission does not comply with the rules given above, you will get a "0" for the whole assignment!**

Collaboration Policy: No collaboration is permitted. Any cheating (copying someone else's work in any form) will result in a grade of -100 for the first offense and -200 for the subsequent attempts.

Grading: Each homework will be graded on the scale 100.

Accompanying PDF file describes the home-brewed, full flavored language **G++**. (Gppsyntax.pdf). We are asking you to develop a lexer that will accept any set of valid g++ expressions and reject incorrect expressions. Your lexer will tokenize the valid statements of the g++ language and identify incorrect statements and produce an error message regarding the incorrect statement.

You are asked to implement the lexer in **two different ways**:

- There are tools to implement lexers given the rules in a meta-grammar such as CFGs. One such tool is "**Flex**" that lets you generate C code to do the lexical analysis. In the first part of the assignment you will use Flex to generate a valid lexer for G++.
- In the second part, you will be implementing a lexer for G++ in Lisp. For this part you are not expected to use a meta-grammar to define the lexical syntax of your language.

Both lexers should start the interpreter. It will read one line at a time from the user and check if the input lexically correct while generating the tokens for later processing.

Part 1. G++ Language Lexer using Flex: Implement your lexer using Flex. Hand in your "gpp_lexer.l" file for this part of the assignment . You are also expected to submit the corresponding C file generated by flex in "gpp_lexer.c" along with any other .h or .c file that is needed to compile "gpp_lexer.c" using GCC on Ubuntu (16 or later).

Part 2. G++ Language Lexer in Lisp. Implement your lexer in Common Lisp. Hand in your "gpp_lexer.lisp" file for this part of the homework. This file should have a function called "gppinterpreter" that will start your interpreter. "gppinterpreter" can have zero or one input. The input can be a file name which will be loaded by the interpreter and interpreted right away.

Important note!

Full score for each part requires the lexer code to implement the proper regular expression or DFA for keywords, identifiers as well as integer values. You may not use available Common Lisp code for regular expression finding. A penalty will be applied for those not implementing a proper DFA or regular expression reader.