# CSE341 – Programming Languages (Fall 2024)

# Homework #2

**Hand-in Policy**: Source code should be handed in via Teams. No late submission will be accepted.

**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 ones.

**Grading**: Each project will be graded on the scale 100.

---

**G++ Language Lexer**: Given the description of the G++ language (G++Syntax.pdf) you are asked to implement a lexer that does the tokenization of any set of valid G++ expressions or statements. Lexical analysis rules of G++ is explained in the first 4 pages of G++Syntax.pdf. Lexically incorrect statements or expressions should be identified, and an error should be generated.

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

1. 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.
2. For this course, you will be implementing a lexer for G++ in Lisp. For this 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.

**G++ Language Lexer using Flex** (35 points): Implement your lexer using Flex. Hand in your "gpp_lexer.l" file for this part of the homework. 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).

Full score would require the lexer code to implement the proper regular expression or DFA for keywords, identifiers as well as integer values. 20 points will be taken away for those not implementing a proper DFA or regular expression reader.

**G++ Language Lexer in Lisp** (65 points): 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.

Full score would require 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. 20 points will be taken away for those not implementing a proper DFA or regular expression reader.

**Examples:** The following table provides the correct tokenization for a few instances of G++ language. Note that lexical syntax error messages should give some information about the error encountered.

| | |
|---|---|
| ;; hello.g++<br>(deffun sumup (x)<br>;; function body<br>(if (equal x 0)<br>1<br>(+ x (sumup (- x 1)))<br>)<br>) | COMMENT<br>OP_OP<br>KW_DEFFUN<br>IDENTIFIER<br>OP_OP<br>IDENTIFIER<br>OP_CP<br>COMMENT<br>OP_OP<br>.... |
| (list 1 2 123f12) | OP_OP<br>KW_LIST<br>VALUEI<br>VALUEI<br>VALUEF<br>OP_CP |
| (liste 1 2 3) | OP_OP<br>SYNTAX_ERROR liste cannot be tokenized |

Handin your code in a single tar for zip file named yourstudentnumber_lastname_firstname_hw2.zip/rar. For example, if your name us John Wayne, student number is 123456789 and zipped everything in a RAR file, then you should submit your file as “123456789_Wayne_John.zip”.