CSE341 – Programming Languages (Fall 2018)
Project #1

Handed out: 13:00 Monday November 12, 2018.

Due: 11:55pm Wednesday November 28, 2018.

Hand-in Policy: Source code should be handed in via Moodle with file name "student_id.cl". 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 (100 points): Given the description of the G++ language (G++Syntax.pdf) you are asked to implement the lexer that does the tokenization of a given G++ program in a file.

You are expected to submit the file "student_id.cl" with a function called "lexer". This function should take a file name and perform lexical analysis of the program contained within this file. The output of the function should be the tokens in a list. The only output of the program on screen must be content of the list such as "(this is a sample output)".

Grading: Full score would require the lexer code to implement the proper regular expression or DFA for 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. Any submission which has a syntax or runtime error results in 0 grade.

Sample Input:

```
(deffun sumup (x)
        (if (equal x 0)
                1
                (+ x (sumup (- x 1)))
        )
)
```

Sample Output: (There may be mistake(s))

```
(("operator" "(") ("keyword" "deffun") ("identifier" "sumup") ("operator" "(") ("identifier" "x")
("operator" ")") ("operator" "(") ("keyword" "if") ("operator" "(") ("keyword" "equal") ("identifier" "x")
("integer" "0") ("operator" ")") ("integer" "1") ("operator" "(") ("operator" "+") ("identifier" "x")
("operator" "(") ("identifier" "sumup") ("operator" "(") ("operator" "-") ("identifier" "x") ("integer" "1")
("operator" ")") ("operator" ")") ("operator" ")") ("operator" ")") ("operator" ")")
```