CSC345-01 Assignment #1 (version 1.1)

```
Original Due:  4:25 PM, Tuesday, September 20
    Extended:  4:25 PM, Tuesday, September 27
```

You must complete this assignment by yourself. You cannot work with anyone else in the class or with someone outside of the class. You may not copy solutions from the world wide web. The code you write must be your own.

Provided Files:

- A1.java - A shell file
- i1.txt - A sample input file
- i2.txt - A sample input file
- i3.txt - A sample input file
- i4.txt - A sample input file
- i5.txt - A sample input file
- w1.txt - The required output for i1.txt (Mac/Linux users should use m1.txt)
- w2.txt - The required output for i2.txt (Mac/Linux users should use m2.txt)
- w3.txt - The required output for i3.txt (Mac/Linux users should use m3.txt)
- w4.txt - The required output for i4.txt (Mac/Linux users should use m4.txt)
- w5.txt - The required output for i5.txt (Mac/Linux users should use m5.txt)

Description: Implement a lexical analyzer in Java for a small language, called **AC** for **A**dding **C**alculator, that accommodates two forms of numerical data types, allows computation and printing of numerical values, and offers a small set of variable names to hold the results of computations. Below are the language **AC**'s tokens and lexemes:

Token ID	**Token Name**	**Lexeme in Regular Expression**
0	FLOATDCL	f
1	INTDCL	i
2	PRINT	p
3	ID	a+b+c+d+e+g+h+j+k+l+m+n+o+q+r+s+t+u+v+w+x+y+z
4	ASSIGN	=
5	PLUS	+
6	MINUS	-
7	INUM	$(0+1+2+3+4+5+6+7+8+9)^+$
8	FNUM	$(0+1+2+3+4+5+6+7+8+9)^+.(0+1+2+3+4+5+6+7+8+9)^+$

The lexical analyzer shall read a source program written in **AC** as a text file and prints out a stream of lexemes, where each lexeme represents an instance of some symbols in the **AC**'s alphabet. For example, strings such as 5 and 3.2 are recognized as lexemes of token types `INUM` and `FNUM`, respectively. Reserved keywords such as `f` and `p` are distinguished from identifiers such as `a` and `b`. The lexical analyzer shall ignore comments and symbols that serve only to format the source code, such as blanks (' '), tabs ('\t'), and new lines ('\n'). The lexical analyzer should terminate right after it reads any undefined or invalid symbol (see i5.txt and w5.txt).

Given the input file, your program should produce exactly same required output. Even minor differences in output will cause you to fail grading tests and lose points. You are highly recommended to use a diff tool to ensure that your program produces the correct outputs. To validate your code, you will need to create more non-trivial inputs for testing on your own.

For this assignment, you are NOT allowed to use any member methods of class java.util.**Scanner** except **hasNextLine** and **nextLine**. For example, you CANNOT use **hasNextInt**, **nextInt**, **hasNextFloat**, **nextFloat**, **hasNextDouble**, and **nextDouble** for this assignment.

You must use the provided shell file (**A1**.java) for this assignment.

Submission: **A1**.java

General Programming Assignment Requirements:

- Classes must be in the default (no package statement) unless otherwise specified. You will lose all points if you put a package statement in your program.
- If your program that does not compile, you will lose all points.
- If you submit the wrong file, you will lose all points.
- You must add the header and fill it in the shell file. Otherwise, you will lose all points.

Checklist: Did you remember to:

- worked on the programming assignment by yourself?
- fill in the header in your file **A1**.java?
- ensure your program does not suffer a compile error or runtime error?
- ensure your program creates the correct output and that it matches the expected output exactly?
- put a comment at the start of each method describing in broad terms, what it does?
- properly indent your source code so that your indenting is readable and consistent?
- use good names for variables to make your program easy to understand?
- turn in your Java source code in a file named **A1**.java and your own input files through D2L?