Comp 333 Programming Project # 1 Part A ( 20 pts)

Due: Sept 26

**Create a scanner for the Calculator Language**

The Extended BNF for the Calculator Language was discussed in class. We also discussed an algorithm (pseudocode) to create a token list in a Calculator Language program. (See class power points for Chap 2 Part 2). Create a scanner to create a token list in a Calculator Language program. Use the scanner pseudocode from the Chap 2 Part 1 power points. Your program should be written in java. Your program should not check syntax. Thoroughly test your program.

**Sample Input:**

“ read radius ; area = 3.1415 \* radius; write area .”

**Sample Output ( This is the token list)**

read

id: radius

semicolon

id: area

assign

number: 3.1415

op: \*

id: radius

semicolon

write

id: area

period

**Sample Input: (program does not have to be syntactically correct)**

“x + y = 18 ; ( 78 - a”

**Sample Output:**

id: x

op: +

id: y

assign

number: 18

semicolon

lparen

number: 78

op: -

id: a

**Sample Input:** “ x = w % 10 . “

**Sample Output:**

id: x

assign

id: w

Error: Unknown symbol: %

Hand in:

1. Submit source code (one java file called “CalcLangScanner.java” with your name and date embedded in file) to Moodle. Instructor will test your code with random test cases.
2. Hard copy of source code with instructor test cases runs and results to be turned in class. Instructor Test Cases will be posted on Sept 20. If your program is only partially working, you must also submit a typed explanation of what works and what doesn’t work and, if possible, include an example of input where your program works.

**Extended BNF Grammar for the Calculator Language**

<progr> 🡪 <stmt-lst> .

<stmt-lst> 🡪 <stmt> { ; <stmt>}\*

<stmt> 🡪 <read-stmt> | <write\_stmt> | <assign\_stmt>

<read-stmt> 🡪 read <id>

<write-stmt> 🡪 write <id>

<assign-stmt> 🡪 <id > = <expr>

<expr> 🡪 <expr> { <op> <expr>}\* | ( <expr> ) | <number> | <id>

<op> 🡪 + | - | \* | /

<id> 🡪 <letter> { <letter > | <digit> }\*

<number> 🡪 <digit> { <digit> }\* [. { <digit> }\*]