CSC311-01 Theory of Computation

Programming Homework A

Pretending that you were programing I or II students, write a program to identify all valid numerical literals, without using any other problem-solving tools such as DFA or regular expression, etc.

In C++ programming language, all the following expressions represent valid numerical “literals”:

3
13.
.328
41.16
+45.80
+0
-01
-14.4
1e12
+1.4e6
-2.e+7
01E-06
-.4E-7
00e0
+0e1
.2E-03

Literals consist of an integral part and an optional exponent part:

- Integral part consists of an optional sign (‘+’ or ‘-‘) and literal value.
    - Literal value should consist of at least one digit either before or after the decimal point.
    - However, the decimal point is also optional.
- Exponent part (which is optional) starts with either ‘e’ or ‘E’, and if it appears, the number following it (‘e’ or ‘E’) must be an integer (with optional ‘+’ or ‘-‘ sign).

Assume that there are no limits on the number of consecutive digits in any part of literal (but the length of string is limited to 50 digits/characters.)

Write a C++ program that read in the filename, which is a text file of strings, one string on each line, and output the result to both the screen AND a text file (*the output file name should be based on the input file name*), each particular string whether each string is a valid numeric literal or not with **correct alignment**. **Your program should account for all possible user input.**

Your program should execute interactively on your computer. Please test your program thoroughly and make sure it is free of errors. Points will be deducted for each error that I discover for you.

Please also pay attention to program design and follow all the good programming practice. I reserve the right to deduct points based on these criteria.

Submit:

- *Program design*
- *Hierarchical diagrams/ Structure charts*
- *Procedure specification*
- **Program listing**
- **Testing files/cases with Justification**
- **Results of test files/cases**

Example input:

PA_testfile.txt:

```
3
13.
.328
41.16
+45.80
+0
-01
#
```

Example output: (note that your output file name should be based on the input file name)

PA_testfile_output.txt

```
3           is a valid numeric literal
13.         is a valid numeric literal
.328        is a valid numeric literal
41.16       is a valid numeric literal
+45.80      is a valid numeric literal
+0          is a valid numeric literal
-01         is a valid numeric literal
#           is NOT a valid numeric literal
```