Laboratory 3

1. Title of the Laboratory Exercise: Program to count no of:
    a. +positive and –negative integers
    b. +positive and –negative fractions

2. Introduction and Purpose of Experiment

.

2. Aim and Objectives

Aim

Objectives

At the end of this lab, the student will be able to

3. Experimental Procedure

Students are required to carry out the following steps:

- Algorithm
- Write the Lex program
- Compile and execute the program (steps)
- Complete the documentation for the given problem

4. Presentation of Results

main.cpp

```
#include <iostream>

extern int yylex();

int dposcnt = 0, dnegcnt = 0, fposcnt = 0, fnegcnt = 0;

auto main(int argc, char* argv[]) -> int {
    std::cout << "enter different numbers" << '\n';

    if (yylex() == 0)
```

```
        std::cout << "\nparsed successfully" << '\n';
    else
        std::cerr << "error parsing" << '\n';

    std::cout << '\n';

    std::cout << "Positive Decimals : " << dposcnt << '\n'
              << "Negative Decimals : " << dnegcnt << '\n'
              << "Positive Fractions : " << fposcnt << '\n'
              << "Negative Fractions : " << fnegcnt << '\n';

    return 0;
}
```

lab03.l

```
%{

#include <iostream>

extern int dposcnt, dnegcnt, fposcnt, fnegcnt;

%}

%option noyywrap

white       [ \n\t]+
digit       [0-9]
integer     [digit]+
exponent    [eE][+-]?{integer}

%%

\+?{digit}+             { std::cout << "found +ve decimal : " << yytext << '\n'; dpos
cnt++; }
\+?{digit}+\.{digit}*   { std::cout << "found +ve fraction : "<< yytext << '\n'; fpos
cnt++; }
-{digit}+               { std::cout << "found -
ve decimal : " << yytext << '\n'; dnegcnt++; }
-{digit}+\.{digit}*     { std::cout << "found -
ve fraction : "<< yytext << '\n'; fnegcnt++; }
{white}                 { ; } // eat whitespaces
```

Makefile

```
# name of the files and the program
NAME = lab03

# compiler setup
CC = g++
```

```
LEX = flex

all: ${NAME}

${NAME}: main.cpp lex.yy.c
    ${CC} main.cpp lex.yy.c -o ${NAME}

lex.yy.c: ${NAME}.l
    ${LEX} ${NAME}.l

clean:
    rm -f lex.yy.c  ${NAME}

run:
    @./${NAME}
```

5. Analysis and Discussions

```
shadowleaf@SHADOWLEAF-ROG /mnt/d/University-Work/University-Work-SEM-06/Compilers/Lab03
./lab03
enter different numbers
+123.2
found +ve fraction : +123.2
12
found +ve decimal : 12
35.4
found +ve fraction : 35.4
-23
found -ve decimal : -23
-345.5
found -ve fraction : -345.5
parsed successfully
Positive Decimals : 1
Negative Decimals : 1
Positive Fractions : 2
Negative Fractions : 1
shadowleaf@SHADOWLEAF-ROG /mnt/d/University-Work/University-Work-SEM-06/Compilers/Lab03
```

Figure 0-1 OUTPUT

```
count_numbers()

1. for each character in character stream
2.     if regex_match \+?{digit}+
3.         dposcnt++
4.     if regex_match \+?{digit}+\.{digit}*
5.         fposccnt++
6.     if regex_match -{digit}+
7.         dnegcnt++
8.     if regex_match -{digit}+\.{digit}*
```

```
9.          fnegcnt++
10.display dposcnt, fposcnt, dnegcnt, fnegcnt
```

6. Conclusions

NOTE:

A Real Number can be regex matched by

```
-?(([0-9]+)|([0-9]*\.[0-9]+)([eE][-+]?[0-9]+)?)
```

where \. denotes a literal period.

Context Sensitivity. LEX provides some support for contextual grammatical rules.

- If ^ is the first character in an expression, then this expression will only be matched at the beginning of a line.
- If $ is the last character in an expression, then this expression will only be matched at the end of a line.
- If r and s are two LEX regular expressions then r/s is another LEX regular expression.
- It matches r if and only if it is followed by an s.
- It is called a trailing context.
- After use in this context, s is then returned to the input before the action is executed. So the action only sees the text matched by r
- Left context is handled by means of start conditions which we will talk about later.

7. Comments

a. Limitations of Experiments

The experiment does not define the notations to be used for fractional numbers and hence few assumptions were made. Fractions like `.10`, `-.20` are not allowed for this experiment.

b. Limitations of Results

The solution program written for this experiment does not accommodate for that fact that fractional number can also be represented in the exponent form, i.e. $2.5e{-}20$ to represent 2.5×10^{-20}. In simple words scientific notation is not considered in this program. Refer to the Note in Conclusion for a solution regex for matching real numbers.

c. Learning happened

We learnt how to write regex expressions to match numbers and count the positive and negative numbers from a given stream of characters.

d. Recommendations

The experiment should define the decimal and fractional number notations to be used.

Component	Max Marks	Marks Obtained
Viva	6	
Results	7	
Documentation	7	
Total	20	