CS352 Lab 1

CS352 Lab 1. Lexical Analysis

Part 0

Copy the initial files of the project into your home directory .

Login to data.cs.purdue.edu and type

```
    mkdir cs352
    cd cs352
    cp  /homes/cs352/Spring2017/lab1-scanner/lab1-src.tar .
    tar -xvf lab1-src.tar
```

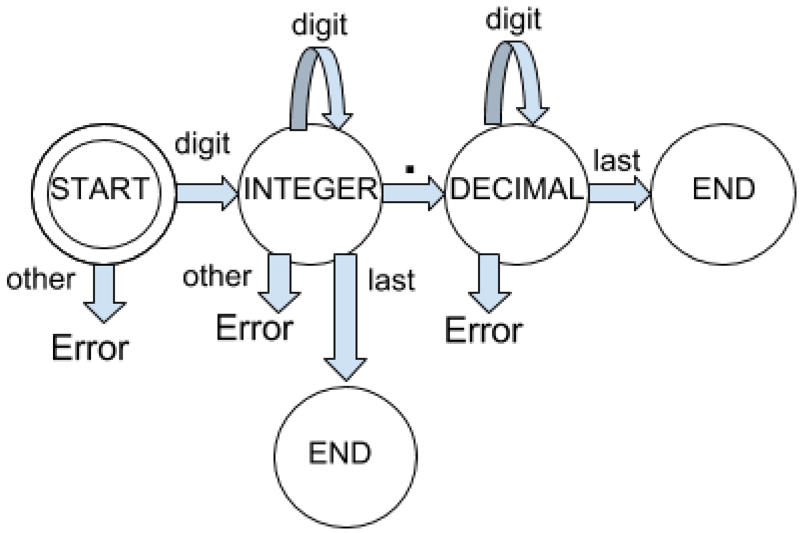
Part 1

DFSA for a decimal number

The regular expression of a number such as 14.67 is given by

[0-9]+.[0-9]*

The DFSA (Deterministic Finite State Automaton) for this regular expression is shown here:



There is a sample implementation of this DFSA for the decimal number in lab1-src/DecimalParser .java. See the source file and become familiar with them.

Type “make” and run the DecimalParser program:

```
$ java DecimalParser
```

```
Usage: java NumberParser value
$ java DecimalParser 1.5
Value=1.5
$ java DecimalParser 0.5
Value=0.5
$ java DecimalParser 0
Value=0.0
$ java DecimalParser 56
Value=56.0
$ java DecimalParser 56.3
Value=56.3
$ java DecimalParser 56.
Value=56.0
java DecimalParser .9
Exception in thread "main" java.lang.Exception: Bad format
        at DecimalParser.MyParseDecimal(DecimalParser.java:37)
        at DecimalParser.main(DecimalParser.java:11)
$ java DecimalParser sd
Exception in thread "main" java.lang.Exception: Bad format
        at DecimalParser.MyParseDecimal(DecimalParser.java:37)
        at DecimalParser.main(DecimalParser.java:11)
```

Now you will use this program as base to implement parsing of a floating point number .

DFSA For a Floating Point Number

The regular expression for a floating point number such as +45.2E-23 is

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

TODO 1:
Draw a DFSA (Deterministic Finite State Automaton) for this regular expression in the computer , convert it to PDF format, and place this file in lab1-src/float.pdf

TODO 2:
Understand the DecimalParser .java implementation and using that file as base implement the DFSA for the floating point number inside the file lab1-src/FloatParser .java

Make sure that the following numbers are accepted:

1.2
-1.5
3e4
2E-90
-1.2e45
-989.455E+20

Part 2

Study the documentation of JavaCC. Here are some links of tutorials that can help you get started:

- http://www.engr.mun.ca/~theo/JavaCC-T utorial/javacc-tutorial.pdf
- http://cs.lmu.edu/~ray/notes/javacc/

The javacc executable is found in data in /homes/cs352/javacc-6.0/bin/javacc and there are some examples that you can see in in /homes/cs352/javacc-6.0/examples. Specifically the examples in /homes/cs352/javacc-6.0/examples/SimpleExamples are useful.

In this lab you will implement the lexical analysis that will take a program in SImpleC and it will extract the tokens.For this part modify the file SimpleC.jj. The SimpleC language is described here.

Your program will print a report such as this one:

```
java SimpleC < tests/hello.c
(VOID, “void”)
(ID,  “main”)
(LPARENT, “(“ )
(RPARENT, “)”)
(LCURLY, “{”)
(ID, “printf”)
(LPARENT, “(“ )
(STRING_CONST, ”Hello world...\n”)
(RPARENT, “)”)
(SEMICOLON, “;”)
(RCURLY, “}”)
 Total Tokens: 11
```

Create a Token expression for every token that is needed for the SimpleC lang uage.

Deadline

The deadline of this project is Monday January 30th at 1 1:59pm. To turnin your project type:

```
cd
cd cs352
turnin -c cs352 -p lab1 lab1-src
```

Make sure that the .git/ is there as well since it is used to verify the evolution of your project.