CS1319: Programming Language Design and Implementation
Department of Computer Science

Assignment 2: Guide

This is a guide on how to complete Assignment 2. Reading this should allow you to familiarise yourself with how you are expected to output tokens, how your directory should look and how you should be compiling your files. Note that this guide uses the Linux version of Flex, and so the commands on Windows or Macintosh might vary a little.

For this guide, we will be using the following lexer named `test.l` as an example:

```
%{
%}

EOL \$
NUMBER [1-9][0-9]*
PUNCTUATOR [()]
OPERATOR [+\-*/%]
WS [ \n\t]

%%

{NUMBER}        {printf("<NUMBER,%s>\n",yytext);}
{PUNCTUATOR}    {printf("<PUNCTUATOR,%s>\n",yytext);}
{OPERATOR}      {printf("<OPERATOR,%s>\n",yytext);}
{EOL}           {printf("<EOL SYMBOL>\n");}
{WS}            ;

%%

int yywrap(){
    return 1;
}
```

According to the specifications of the assignment, the function `main()` should be in a separate `.c` file. The following C file `test.c` is a file that can be used to compile your lexer:

```
int yylex();

int main(){
    yylex();
}
```

Use the following commands in your terminal to compile the code. Do not forget that you need to create a `makefile` as well:

```
flex test.l
gcc lex.yy.c test.c -ll
```

This will create a file called `a.out`. To test your code with a `.nc` file, use the following terminal command:

```
./a.out < test.nc
```

The file `test.nc` (as well as its output when it is passed to the lexer program) is given below:

```
((5*3)+(6/3))$
```

The output of the program is:

```
<PUNCTUATOR,(>
<PUNCTUATOR,(>
<NUMBER,5>
<OPERATOR,*>
<NUMBER,3>
<PUNCTUATOR,)>
<OPERATOR,+>
<PUNCTUATOR,(>
<NUMBER,6>
<OPERATOR,/>
<NUMBER,3>
<PUNCTUATOR,)>
<PUNCTUATOR,)>
<EOL SYMBOL>
```

Feel free to add more lines to this file and take a look at their outputs to further understand what this lexer is doing. Note that in your assignments, the tokens you generate must only be classified into one of the following categories: $keyword$, $identifier$, $constant$, $string - literal$, $punctuator$. The output categories in the example lexer are just that: examples.

Remember to follow all the instructions given in the assignment for full credit. We expect a `.zip`, `.tar` or `.rar` file containing a `.l` file, `.c` file, `makefile`, `.nc` file and a `.pdf` file which explains the working of your lexer and the design choices you have made.