Flex (Fast Lexical Analyzer Generator)

1. Tool used for generating lexical analyzers (scanners or lexers)

2. Flex and Bison both are more flexible than Lex and Yacc and produces faster code.

3. Bison produces parser from the input file provided by the user. (Generation of parse tree)

Note: The function yylex() is the main flex function which runs the Rule Section

```
sudo apt-get install flex
```

Step 1: An input file describes the lexical analyzer to be generated named lex.l is written in lex language.

The lex compiler transforms lex.l to C program, in a file that is always named lex.yy.c.

Step 2: The C complier compile lex.yy.c file into an executable file called a.out.

Step 3: The output file a.out take a stream of input characters and produce a stream of tokens.

Program Structure: (or to write a basic code)

Unlike other programming language here we will have 3 sections.

1. Definition Section:

2. Rules Section

3. User Code Section

1. Definition Section:

The definition section contains the declaration of variables

Syntax:

```
%{
    _____________________
    _____________________
%}
```

Anything written in this brackets is copied directly to the file **lex.yy.c**

2. Rules Section

It contains a series of rules in the form: ***pattern action***

*Pattern **must be unintended** and action begin on the same line in {} brackets*

Ex. pattern === [a-z] any alphabetic letter (in small)

. any character except newline

Syntax:

%%

pattern action

%%

3. User Code Section

This section contain C statements and additional functions

```
int yywrap(){} # wraps the above rule section
int main(){
yylex(); this is the main flex function which runs
the Rule Section*/

}
```

How to run the program:

Step 1: lex filename.l or lex filename.lex depending on the extension file is saved with

Step 2: gcc lex.yy.c

Step 3: ./a.out

Step 4: Provide the input to program in case it is required

Count the number of capital characters in a string

```
      Section 1
%{
int count = 0;
%}
     Section 2
%%
[A-Z] {printf("%s capital letter\n", yytext);
count++;}
. {printf("%s not a capital letter\n", yytext);}
\n  {return 0;}
%%
yytext is the text in the buffer  to print current buffer value
```

Section3 code section

```
int yywrap(){}
int main(){

    yylex();
    printf("\nNumber of Captial letters "in the given input - %d\n",count);
return 0;
}
```

How to compile file

- Lex filename.l
- Gcc lex.yy.c
- ./a.out

For more details: https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/