Flex (Fast Lexical Analyzer Generator)

FLEX is a computer program used to generate lexical analyzers or scanners or lexers written by Vern Paxson in C . It is worked with Berkeley Yacc parser generator or GNU Bison parser generator. Flex and Bison both are more flexible than Lex and Yacc and produces faster code.

Bison produces parser from the input file provided by the user. The function yylex() is automatically generated by the flex when it is provided with a .l file and this yylex() function is expected by parser to call to retrieve tokens from current/this token stream.
The function yylex() is the main flex function which runs the Rule Section and extension (.l or .lex) is the extension used to save the programs.

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 compiler 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:

In the input file, there are 3 sections:

1. Definition Section: The definition section contains the declaration of variables, regular definitions, manifest constants. In the definition section, text is enclosed in “%{ %}” brackets. Anything written in this bracket is copied directly to the file lex.yy.c

Syntax:

```
%{
//
}%
```

2. Rules Section: The rules section contains a series of rules in the form: pattern action and pattern must be unintended and action begin on the same line in {} brackets. The rule section is enclosed in “%% %%”.

Syntax:

```
%%
pattern action
%%
```

3. User Code Section: This section contain C statements and additional functions. We can also compile these functions separately and load with the lexical analyzer.

Basic Program Structure:

```
%{
// Definition
%}
%%
Rules
%%
User Code Section
```

How to run the program:

To run the program, it should be first saved with the extension .l or .lex. But better to save in .l, then run the below commands on terminal in order to run the program file.
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

yywrap(): Function yywrap is called by lex when input is exhausted. Return 1 if you are done or 0 if more processing is required.
yyin(): It scans the stream of characters and try to match the characters with the regular expression provided by user.
yylex() is the function which we have to invoke to start the process.
yylval(): It generates a token value for each token
yyout() : It is a pointer to a file where it has to keep the output

LAB 1:

1. Introduction to LEX. Write briefly about the following:
    a. Structure of Lex File
    b. Compile and Run Steps
    c. Some Basic Lex Functions

2. Basic Lex program to count the number of:
    a. Lines
    b. Words
    c. Capital Letters
    d. Small Letters
    e. Numbers (10,21)