* Assignment No. 6 * (group B)

* Title:- Lexical analysis to generate tokens.

* Problem statement:- Write a program using lex specification to implement lexical analysis phase of compiler to generate tokens of subset of Java program.

* objective:-

* Understand the importance & usage of lex automated tools.
* Appreciate a role of lexical analysis phase in Compilation.

* S/w and H/w apparatus:-

64 bit open source fedora, Eclipse IDE, LEX and YACC.

* Theory:-

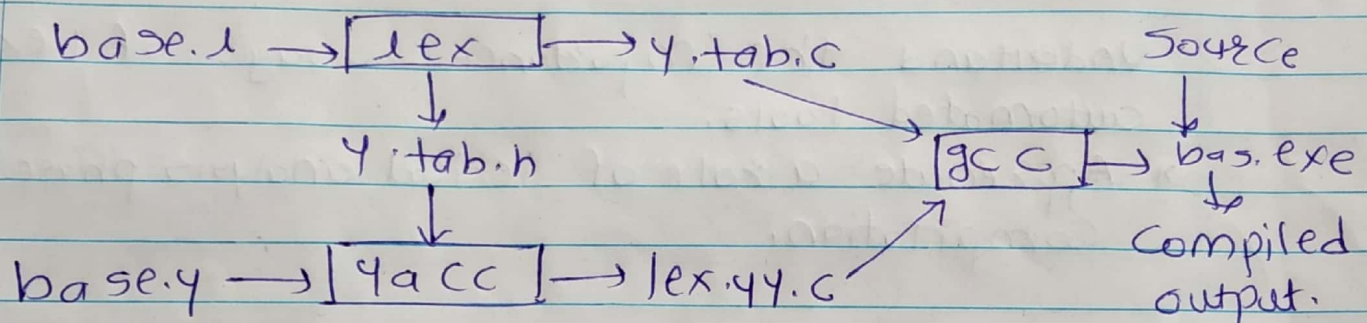
Lex:-

Lex is officially known as "lexical Analyzer". It's main job is to break up an input stream into more usable elements.

e.g. If we are writing program for C language. then lex Analysis is performed to seprate and analyze each words. tokens and operators in C.

YACC:- YACC is also known as "parser". It's job is to analyze the structure of input stream and operate of the "big picture".

YACC stands for "Yet Another Compiler Compiler". This because this kind of analysis of text files is normally associated with writing compilers.

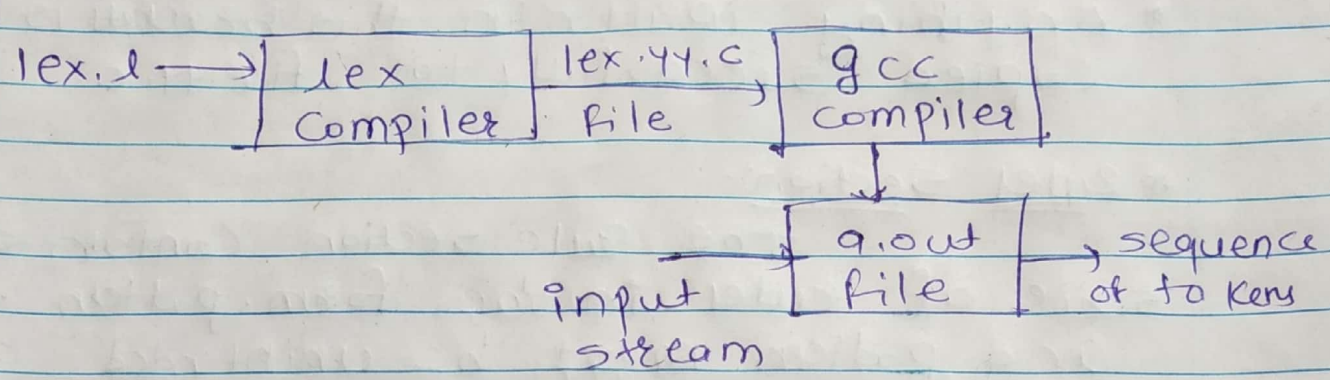


Function of lex:-

① Firstly lexical analyzer creates a program lex.l in the language of lex. Then lex compiler runs the lex.l program and produces a C program lex.yy.c

② Finally C Compiler runs the lex.yy.C program and produces an object program. a.out.

③ a.out is lexical analyzer that transforms an input stream into a sequence of tokens.



Lex file format:-

* A lex program is seperated into 3 sections by %% delimeters.
* The format of lex source is as follows.

```
{ definitions }
%%
{ rules }
%%
{ User Subroutines }
```

* Program structure:-

* Definition section:-

The definition section contains the declaration of variables, regular definitions, manifest constants.

* In the definition sections text is enclosed in

```
%{
%}
```

brackets.

* Anything written in this brackets is Copied successfully to file.yy.c

* Rules Section:-

The rule section Contain 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 "%% %%".

* User Code Section:-

* This section contains C statements & Conditional functions.
* We can also compile these functions seperately and load this with lexical analyzer.

* Advantage of Lexical analysis:-

* Lexical Analyzer method is used by programs like Compilers which can use the parsed data from a programmer's Code to create a compiled binary executable code.
* It is used by web browsers to format and display a web page with the help of parsed data from js, html & CSS.
* A seprate Lexical Analyzer helps you to construct a specialized & potentially more efficient.

* Disadvantages:-

① You need to spend significant time reading the source program & partitioning it in the form of tokens.

② some regular expressions are quite difficult to understand compared to PEG or EBNF rules.

③ More effort is needed to develop and debug the lexes and its token descriptions.

* Steps to run the program:-

① file is saved with extension with .l or .lex

Run below commands on Terminal in order to run the program file

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

Step 2:- gcc lex.yy.c

Step 3:- ./a.out

Step 4:- provide input to program in case it is required.

* Test Case:-

```
import java.io.*;      => Preprocessor
class                  => Keyword
Test                   => Identifier
{                      => Block begin
public                 => Access specifier
```

static ⇒ Keyword
Void ⇒ Return type of function.
main ⇒ Identifier
(⇒ Paranthesis begin
string ⇒ Datatype
args ⇒ Identifier
) ⇒ Paranthesis end
{ ⇒ Block begin
int ⇒ Datatype.
b ⇒ Identifier
; ⇒ Delimeter.
} ⇒ Block ends.
} ⇒ Block ends.

Conclusion:- Thus, implemented lex program successfully to generate tokens for java program using lex specification.