# LEX : LEXICAL ANALYZER GENERATOR

Lex is a Translator.

INPUT : LEX SOURCE program having R.E., to match tokens in Input string.

OUTPUT : A C program lex.yy.c which has the function yylex() which is used for scanning the input for tokens.

LEX SOURCE CODE (PROGRAM) has 3 parts

```
declarations
%%
translation Rules
%%
auxillary procedures
```

## Declarations section

- C code, defined between %{ and %}.

example
```
%{
    #include <stdio.h>
%}
```

• Declarations of variables and Regular Definitions which can be used in the translation rules section.

Example:

```
digit     [0-9]
letter    [A-Za-z]
ident   {letter}({letter}|{digit})*
```

—x—

TRANSLATION Rules

are statements of the form:

```
p1        {action1 }
p2        { action 2 }
:
pn        {action  n }
```

Here p denotes R.E; and the associated action is taken when it matches a lexeme in the input.

Auxillary Procedures

C Language Routines which are called in the action parts.

* LEX always matches the longest possible substring to a R.E pattern.

If 2 Rules MATCH at the same length, Lex will use the one which comes first in the Rules section.

- If no pattern matches the input string, LEX's default action is to copy the INPUT to the OUTPUT. ECHO macro makes this explicit.

Example: vi myprogram.l

```
%{
 #include <stdio.h>
%}
digit      [0-9]
letter     [A-Za-z]
ident    {letter}({letter}|{digit})*
%%
if     { printf("Keyword 'if'\n"); }
while { printf("Keyword 'while'\n"); }
{ident} {printf("identifier\n"); }
 .      ECHO;
%%
```

## Sample Session with Lex

```
$ vi myprogram.l
$ lex myprogram.l
$ cc lex.yy.c -ll -o myprogram
$ myprogram
        or
$ myprogram < infile.c
$ myprogram < infile.c > outfile
$ more outfile
$ cat outfile
```

---

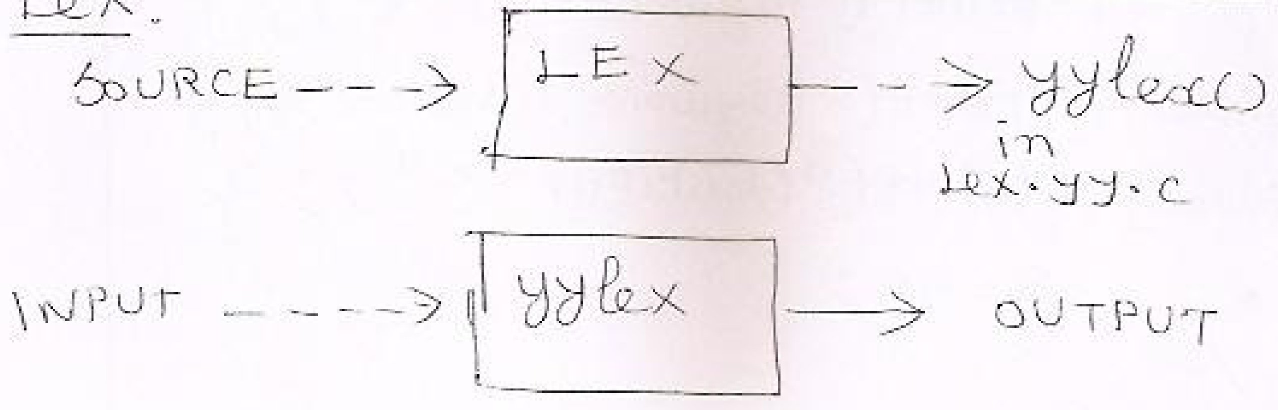
Absolute Minimum LEX program:

```
%%
```

This will copy INPUT to OUTPUT unchanged.

---

Lex:



practice the following Lex programs

```
%%
  /* FileName: bV1.l */
  /* Recognize any entered character
     and echo it */
. | \n     ECHO;
%%
```

```
$ lex bV1.l
$ cc lex.yy.c -ll -o bV1
$ bV1
```

| INPUT | DISPLAY |
|---|---|
| 123 | |
| | 123 |
| asd | |
| | asd |
| . | |
| | . |
| CTRL $\boxed{d}$ ↑ (exit, come to shell) | $ |

Follow these steps for other programs also.

```
/* filename: bv2.l
   To replace all 'a's with
   'A's in the input stream */

%%
"a"      printf("A");
%%
```

```
/* filename: bv3.l */
/* To suppress all spaces and Tabs
   in the input

%%
" "        ;
"\t"       ;
%%
```

```
/* To recognize LC & UC Alphabets */
/* filename: bv4.l */
%%
[^a-zA-Z]      ;
%%
```

```
/* To recognize all NONAlphabets */
/* filename: bv5.l */
%%
[a-zA-Z]      ;
%%
```