Program Code

lex.l

```
%{
    #include<stdio.h>
    #include<stdlib.h>

    int comment=0;
%}

operator [+*\-&/%=!#\[\]]|(<=)|(>=)|[<>]
letter [a-zA-Z]
digit [0-9]
literal (["]({letter}|{operator}|{digit}|[\\n]|[\\t]|[ ])*["])|
{digit}+
identifier ({letter}|_)({letter}|_|{digit})*

%%
\/\/.*   ;
\/\*.* ;
.*\*\/.*  ;
void|main|include|define|printf|scanf|fgets|for|while|int|char|
strlen|FILE|fopen|feof|NULL|if|return|double|continue|break|
strcmp|strcat|fflush|fscanf|fprintf|strcpy|return {printf("\n%s,
keyword",yytext);}
{operator} {printf("\n%s, operator",yytext);}
"1","2","3","4","5","6","7","8","9","0" {printf("\n%s,
literal",yytext);}
{literal} {printf("\n%s, literal",yytext);}
"{"|"}"|"("|")"|";"|","|"." {printf("\n%s, seperator",yytext);}
{identifier} {printf("\n%s, identifier",yytext);}
%%

int yywrap(){}
void main()
{
    yyin=fopen("test.c", "r");
    yylex();
}
```

Test.c

```
#include<stdio.h>
#include<stdlib.h>

/*This is an implementation of lexical analyser using the lex
tool. This program was implemented for the compiler lab*/

void main()
{
    int a,b,c;
    a=5;
    b=8;
    c=a+b*a;
    printf("\nc = %d\n",c);
}
```

Output

```
students@pgcse-HP-280-G1-MT:~/Desktop/R7_66/R7_66/2/1$ lex
lexical.l
students@pgcse-HP-280-G1-MT:~/Desktop/R7_66/R7_66/2/1$ cc lex.yy.c
-o lex
students@pgcse-HP-280-G1-MT:~/Desktop/R7_66/R7_66/2/1$ ./lex

#, operator
include, keyword
<, operator
stdio, identifier
., seperator
h, identifier
>, operator

#, operator
include, keyword
<, operator
stdlib, identifier
., seperator
h, identifier
>, operator



void, keyword
main, keyword
(, seperator
), seperator

{, seperator

int, keyword
a, identifier
,, seperator
b, identifier
,, seperator
c, identifier
;, seperator

a, identifier
=, operator
5, literal
;, seperator

b, identifier
=, operator
8, literal
;, seperator

c, identifier
=, operator
a, identifier
+, operator
b, identifier
*, operator
a, identifier
;, seperator
```

```
printf, keyword
(, seperator
"\nc = %d\n", literal
,, seperator
c, identifier
), seperator
;, seperator

}, seperator
```