```
 //Implementation of Lexical Analyzer using Lex tool
%{
int COMMENT=0;
%}
identifier [a-zA-Z][a-zA-Z0-9]*
%%
#.* {printf("\n%s is a preprocessor directive",yytext);}
int |
float |
char |
double |
while |
for |
struct |
typedef |
do |
if |
break |
continue |
void |
switch |
return |
else |
goto {printf(" kwd");}
"/*" {COMMENT=1;}{printf("comment");}
\+ {if(!COMMENT)printf(" op-plus");}
\- {if(!COMMENT)printf(" op-sub");}
\* {if(!COMMENT)printf(" op-mul");}
\/ {if(!COMMENT)printf(" op-div");}
{identifier}\( {if(!COMMENT)printf("fun");}
\{  {if(!COMMENT)printf("block begins");}
\}  {if(!COMMENT)printf("block ends");}
{identifier}(\[[0-9]*\])? {if(!COMMENT) printf(" id");}

\".*\" {if(!COMMENT)printf("str");}
[0-9]+ {if(!COMMENT) printf("num");}
\)(\:)? {if(!COMMENT)printf("\n\t");ECHO;printf("\n");}
\( ECHO;
= {if(!COMMENT)printf(" op-equ");}
\<= |
\>= |
\< |
== |
\> {if(!COMMENT) printf("rel-op");}

%%


int main(int argc, char **argv)
{
FILE *file;
file=fopen("input.c","r");
if(!file)
{
printf("could not open the file");
exit(0);
}
yyin=file;
yylex();
printf("\n");
return(0);
}
int yywrap()
{
```

```
return(1);
}
```

Open

*input.c
~/Compiler-Lab/2)lexicalanalyzer-lex

```
int a,b,c;

a=b+c;
```

```
akhil@Ubuntu:~/Compiler-Lab/2)lexicalanalyzer-lex$ lex lexicalanalyzer-lex.l
akhil@Ubuntu:~/Compiler-Lab/2)lexicalanalyzer-lex$ gcc lex.yy.c
akhil@Ubuntu:~/Compiler-Lab/2)lexicalanalyzer-lex$ lex lexicalanalyzer-lex.l
akhil@Ubuntu:~/Compiler-Lab/2)lexicalanalyzer-lex$ gcc lex.yy.c
akhil@Ubuntu:~/Compiler-Lab/2)lexicalanalyzer-lex$ ./a.out
 kwd  id, id, id;

 id op-equ id op-plus id;

akhil@Ubuntu:~/Compiler-Lab/2)lexicalanalyzer-lex$ ls
```