EX. NO.: **ROLL NO.:** 210701295
DATE:

LEXICAL ANALYZER USING LEX TOOL

AIM:

To create a lexical analyser using lex tool.

ALGORITHM:

Step 1: In the headers section declare the variables that is used in the program including header files if necessary.

Step 2: In the definitions section assign symbols to the function computations we use along with REGEX expressions.

Step 3: In the rules section assign each function to their expressions.

Step 4: In the definition section increment the values accordingly to the arithmetic functions respectively.

Step 5: Define cases for different computations.

Step 6: Define the main () and yywrap() function.

SOURCE CODE:

```
%option noyywrap
letter [a-zA-Z] digit
[0-9]
id [_|a-zA-Z]
AO [+|-|/|%|*]
RO [<|>|<=|>=|==]
pp [#]
%{
int n=0;
%}
```

```
%%
“void” printf(“%s return type\n”,yytext);
{letter}*[(][)] printf(“%s Function\n”,yytext);
“int”|”float”|”if”|”else” printf(“%s keywords\n”,yytext);
“printf” printf(“%s keywords\n”,yytext);
{id}({id}|{digit})* printf(“%s Identifier\n”,yytext);
{digit}{digit}* printf(“%d Numbers\n”,yytext);
{AO} printf(“%s Arithmetic Operators\n”,yytext);
{RO} printf(“%s Relational Operators\n”,yytext);
{pp}{letter}*[<]{letter}*[.]{letter}[>] printf(“%s processor
Directive\n”,yytext);
[\n] n++;
“.”|”,”|”}”|”{“|”;” printf(“%s others\n”,yytext);
%%
int main()
{
yyin=fopen(“sample.c”,”r”);
yylex();
printf(“No of Lines %d\n”,n);
}

/*Input*/
#include<conio.h>
void main()
{
int a=b+1c;
}
```

OUTPUT:

```
UKESHWARAN295 :~$ ./a.out
#include<conio.h>
void main()
{
  #include<conio.h> processor Directive
void return type
 main()  Function
{ others
      int a,b,c;
}
        int keywords
 a Identifier
, others
b Identifier
, others
c Identifier
; others
} others
```

RESULT: