

Compiler Construction

D2

Lab1

19BCE248

AIM: To implement lexical analyzer to recognize all distinct token classes: use flex/lex tool to recognize all distinct token classes (Data type, Identifier, constant (Integer, Float, Char, String), Operator (Arithmetic, Relational, Assign, Unary +/-, Increment), Single line/Multi-line comments, Special symbol (;,{ } ())) .

Lex File:

```
%{
int lc=0;
%}
%%

("/" * "(" [^*] | "\" * "/" ) * "*" "/" {printf("Multiline Comments\n");}
[#].* {printf("Header\n");}
[0-9]+/(;"|'|"\\") {printf("Integer ");}
("int"|"float"|"char"|"double"|"struct"|"if"|"while"|"do"|"printf"|"else"|"return"|"for") {printf("Keywords ");}
([0-9]+.[0-9]+) {printf(" Float ");}
([_A-Za-z]([_A-Za-z][0-9])*) {printf("Identifier ");}
([0-9]+[a-zA-Z][a-zA-Z0-9]+) {printf("Invalid ");}
([-+\\ *%]|"++"|"--") {printf(" Arithmetic Operator ");}
(">"|"<"|"=="| ">="|"<="|"!=") {printf(" Relational Operator ");}
[=] {printf("Assignment\n");}
([/]{2}).* {printf("Comments\n");}
("\\n") {lc++;printf("\\t\\t\\t%d\\n",lc);}
%%

int yywrap(){}

int main(){
    yylex();
```

```

    return 0;
}

C File:

#include<stdio.h>

//My First Code

void main(){

    int 1ab=10;

        /*Multiline Comment 123 'a'*/

        Second Line

        */

for(int i=0;i<3;i++){

    a++;

}

    printf("test",'?');

    return 0;

    /*****

    end of

    code 1

    */

}

```

Output:

```

Header
Comments
Identifier Identifier (){
Keywords Invalid Assignment
Integer ;
Multiline Comments
Identifier Identifier
Arithmetic Operator /
Keywords ( Keywords Identifier Assignment
Integer ;Identifier Relational Operator Integer ;Identifier Arithmetic Operator ){
Identifier Arithmetic Operator ;
}
Keywords ("Identifier ","?');
Keywords Integer ;
Multiline Comments
}

```

