Practical No: 1

Name : Patel Savankumar P.
Enroll No : 19BCE519
Subject : Compiler Construction

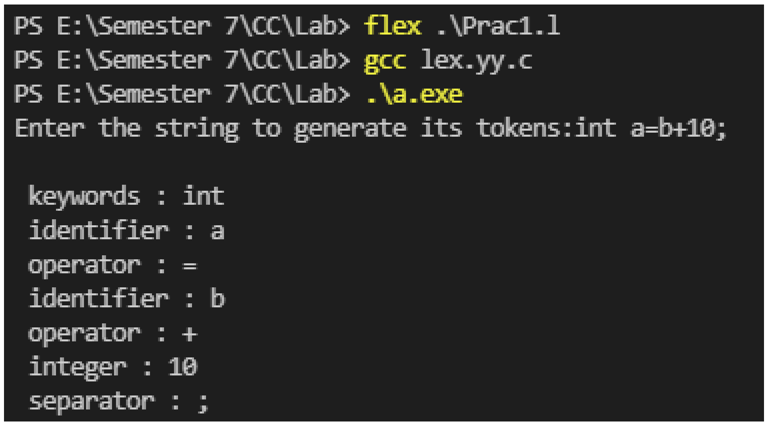
AIM: To implement lexical analyser to recognize all distinct token classes.

Code:

```
%{
    #include<stdio.h>
    #include<string.h>
    int n=0;
%}

%%


"while"|"if"|"else"|"return"|"break"|"case"|"for"|"NULL"|"struct"|"switc
h"|"continue"|"do"|"default" {n++; fprintf(yyout,"\n keywords : %s", yyt
ext);}
"int"|"float"|"bool" {n++; fprintf(yyout,"\n keywords : %s", yytext);}

[a-zA-Z_][a-zA-Z0-
9_]* {n++; fprintf(yyout,"\n identifier : %s", yytext);}
[\+|-
|\*|\/|=|>|<|>=|<=|&|\||%|!|\^|\(|\)] {n++; fprintf(yyout,"\n operator :
 %s", yytext);}
[(){}|,;]    {n++; fprintf(yyout,"\n separator : %s", yytext);}
[0-9]*"."[0-9]+ {n++; fprintf(yyout,"\n float : %s", yytext);}
[0-9]+ {n++; fprintf(yyout,"\n integer : %s", yytext);}
```

```
.    ;

%%

int main()
{
    printf("Enter the string to generate its tokens:");
    yylex();
    printf("\nTotal Identifiers ==> %d",n);
    fclose(yyout);
    return 0;
}

int yywrap()
{
    return 0;
}

int yyerror()
{
    return 0;
}
```

Output:



```
PS E:\Semester 7\CC\Lab> flex .\Prac1.l
PS E:\Semester 7\CC\Lab> gcc lex.yy.c
PS E:\Semester 7\CC\Lab> .\a.exe
Enter the string to generate its tokens:int a=b+10;

 keywords : int
 identifier : a
 operator : =
 identifier : b
 operator : +
 integer : 10
 separator : ;
```

Conclusion:

From this practical I learned how to create a lexical analyzer for any grammar.