Exp. No.: 7 Token Separation using Flex Tool

**AIM:**

Write a Lex program to implement Lexical Analyzer using LEX or FLEX (Fast Lexical Analyzer). The program should separate the tokens in the given C program and display with appropriate caption.

**LEX PROGRAM: (token.l)**

digit [0-9]

letter [A-Za-z]

%{

int count\_id,count\_key;

%}

%%

(stdio.h|conio.h) { printf("%s is a standard library\n",yytext); }

(include|void|main|printf|int) { printf("%s is a keyword\n",yytext); count\_key++; }

{letter}({letter}|{digit})\* { printf("%s is a identifier\n", yytext); count\_id++; }

{digit}+ { printf("%s is a number\n", yytext); }

\"(\\.|[^"\\])\*\" { printf("%s is a string literal\n", yytext); }

.|\n { }

%%

int yywrap(void) {

return 1;

}

int main(int argc, char \*argv[]) {

yyin = fopen(argv[1], "r");

yylex();

printf("number of identifiers = %d\n", count\_id);

printf("number of keywords = %d\n", count\_key);

fclose(yyin);

}

**INPUT SOURCE PROGRAM: (sample.c)**

#include<stdio.h>

void main()

{

int a,b,c = 30;

printf("hello");

}

**Compilation & Execution of Lex Program:**

1. Open Command prompt and switch to your working directory where you have stored your lex file (“.l“).
2. Let lex file be “token.l”. Now, follow the preceding steps to compile and run lex program.

For Compiling **Lex** file:

* + 1. flex token.l
    2. gcc lex.yy.c
  1. For **Executing** the Program
     1. a.exe sample.c

**OUTPUT:**

G:\lex>flex token.l

G:\lex>gcc lex.yy.c

G:\lex>a.exe sample.c

include is a keyword

stdio.h is a standard library

void is a keyword

main is a keyword

int is a keyword

a is a identifier

b is a identifier

c is a identifier

30 is a number

printf is a keyword

"hello" is a string literal

number of identifiers = 3

number of keywords = 5

G:\lex>