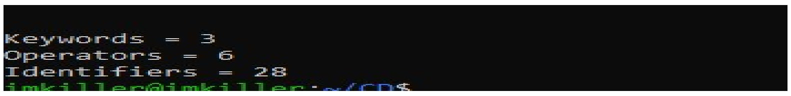
Q12. Write a lex program to count number of identifiers, keywords, operators and separators.

```
%{
        #include <stdio.h>
        int key=0, id=0, op=0;
%}

KEY auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|long|
register|return|short|signed|sizeof|static|struct|switch|typedef|union|unsigned|void|volatile|while

%%
{KEY}+ {printf("Keyword = %s\n", yytext); key++;}
(\+|-|\*|\/|\%)+ {printf("Operator = %s\n", yytext); op++;}
(_?[a-zA-Z])+[a-zA-Z0-9]* {printf("Identifier = %s\n", yytext); id++;}
. {;}
%%

int main()
{
        yylex();
        printf("Keywords = %d\nOperators = %d\nIdentifiers = %d\n", key, op, id);
        return 0;
}
```

<u>OUTPUT:</u>

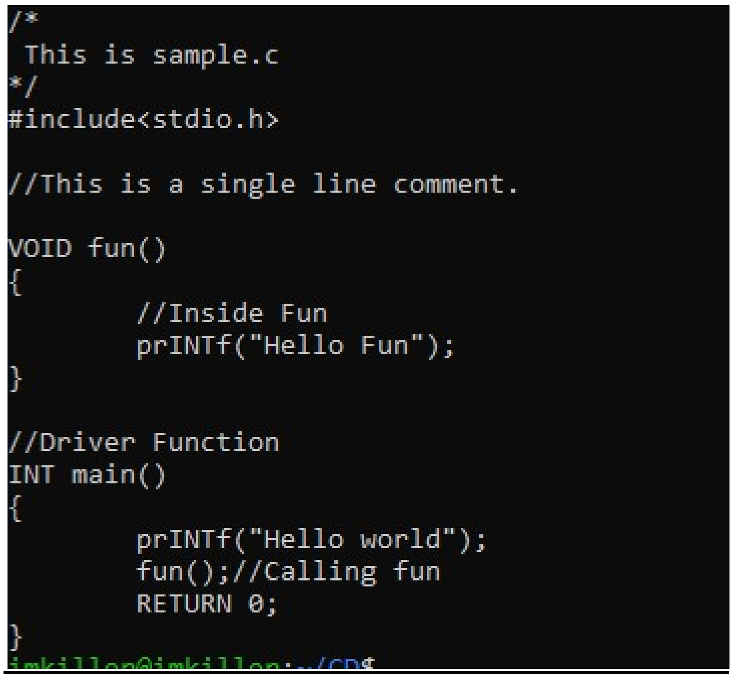


```
Keywords  = 3
Operators  = 6
Identifiers  = 28
imkiller@imkiller:~/CD$
```

Q13. Write a lex program to convert all keywords to uppercase.

```
%{
        #include <stdio.h>
        #include <ctype.h>
%}
KEY auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|long|
register|return|short|signed|sizeof|static|struct|switch|typedef|union|unsigned|void|volatile|while
%%
{KEY} {
int i=0;
for(i=0;i<yyleng;i++)
printf("%c", toupper(yytext[i]));
}
%%

int main()
{
        yylex();
        return 0;
}
```

<u>OUTPUT:</u>

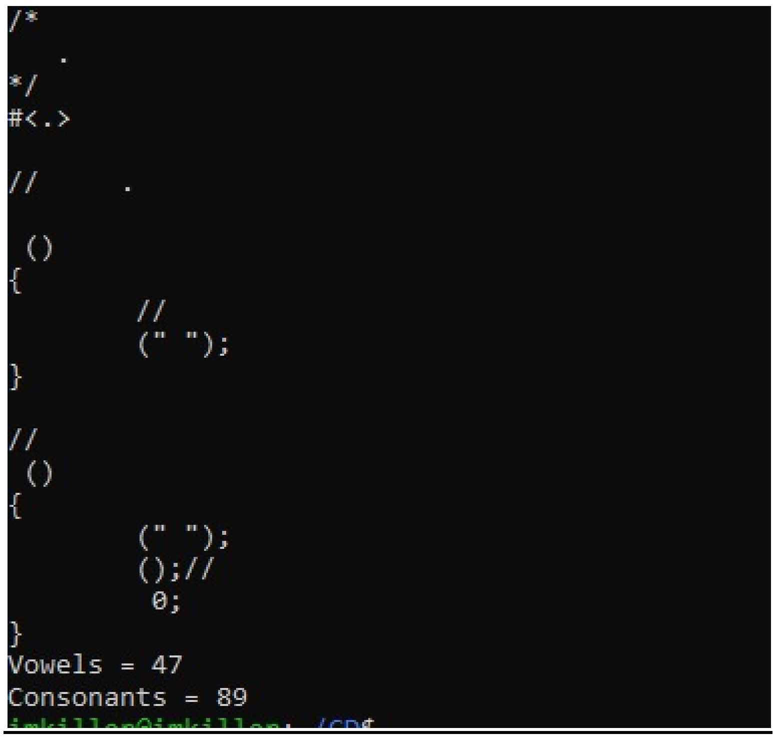


```
/*
 This is sample.c
*/
#include<stdio.h>

//This is a single line comment.

VOID fun()
{
        //Inside Fun
        prINTf("Hello Fun");
}

//Driver Function
INT main()
{
        prINTf("Hello world");
        fun();//Calling fun
        RETURN 0;
}
```

Q14. Write a lex program to count vowels and consonants.

```
%{
        #include <stdio.h>
        int vow=0, con=0;
%}
VOW a|e|i|o|u|A|E|I|O|U
CON b|c|d|f|g|h|j|k|l|m|n|p|q|r|s|t|v|w|x|y|z|B|C|D|F|G|H|J|K|L|M|N|P|Q|R|S|T|V|W|X|Y|Z

%%
{VOW} {vow++;}
{CON} {con++;}
%%

int main()
{
        yylex();
        printf("Vowels = %d\nConsonants = %d\n", vow, con);
        return 0;
}
```

OUTPUT:

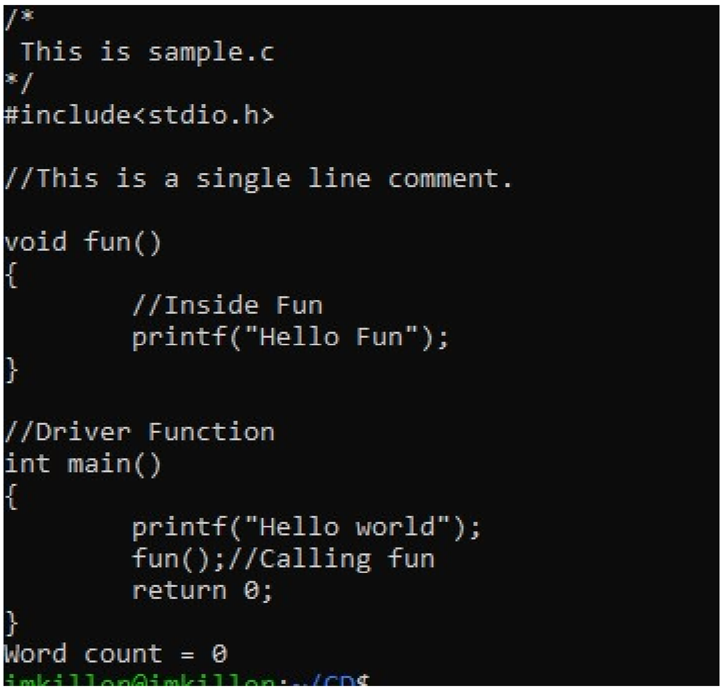


```
/*
   .
*/
#<.>

//     .

 ()
{
        //
        (" ");
}

//
 ()
{
        (" ");
        ();//
         0;
}
Vowels = 47
Consonants = 89
```

Q15. Write a lex program to count all words ending with “ab”.

```
%{
        #include <stdio.h>
        int ct = 0;
        int xc = 0;
%}

%%
[a-zA-Z]*(ab) {
if(xc < 1)
{
xc++;
printf("\n");
}
printf("Word = %s\n", yytext); ct++;
}
%%

int main()
{
        yylex();
        printf("Word count = %d\n", ct);
        return 0;
}
```

OUTPUT:

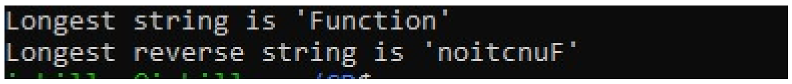


```
/*
 This is sample.c
*/
#include<stdio.h>

//This is a single line comment.

void fun()
{
        //Inside Fun
        printf("Hello Fun");
}

//Driver Function
int main()
{
        printf("Hello world");
        fun();//Calling fun
        return 0;
}
Word count = 0
jmkillen@jmkillen:~/CD$
```

Q16. Write a lex program to reverse the longest word in string.

```
%{
        #include <stdio.h>
        #include <string.h>
        char* longest;
%}
longest [a-zA-Z]+
%%
{longest} {
if (yyleng > strlen(longest))
{
        longest = (char*)realloc(longest, yyleng + 1);
        strcpy(longest, yytext);
}
}
<*>.|\n /* skip all unrecognized text */
%%

int main()
{
        longest = (char*)malloc(1);
        longest[0] = '\0';
        yylex();
        printf("Longest string is "%s'\n", longest);
        int n = strlen(longest);
        char* rlongest = (char*)malloc(n+1);
        int i=0;
        for(i=0;i<n;i++)
        rlongest[i] = longest[n-1-i];
        rlongest[n] = '\0';
        printf("Longest reverse string is "%s'\n", rlongest);
        free(longest);
        free(rlongest);
        return 0;
}
```

OUTPUT:

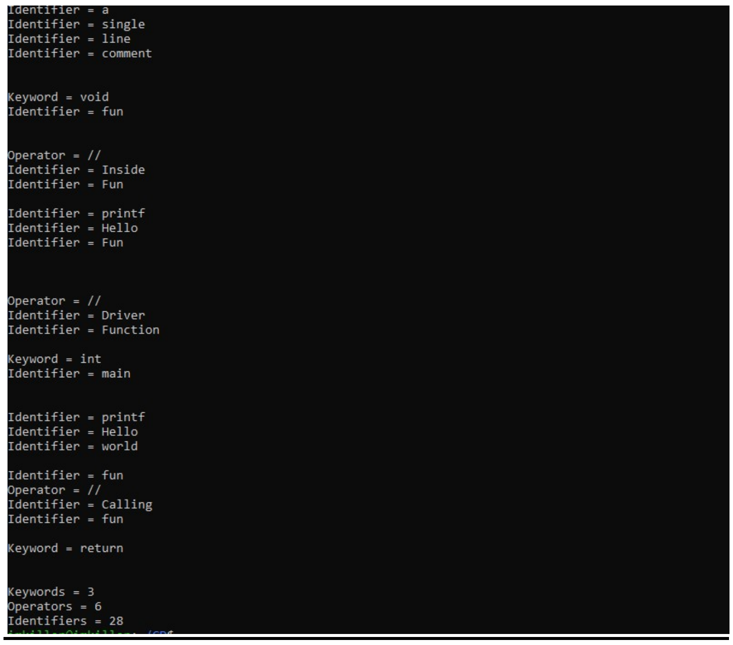


```
Longest string is 'Function'
Longest reverse string is 'noitcnuF'
```

Q17. Write a lex program to count keywords (even having underscore at start), identifiers, operators and separators.

```
%{
	#include <stdio.h>
	int key=0, id=0, op=0;
%}

KEY auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|long|
register|return|short|signed|sizeof|static|struct|switch|typedef|union|unsigned|void|volatile|while

%%
{KEY}+ {printf("Keyword = %s\n", yytext); key++;}
(\+|-|\*|\/|\%)+ {printf("Operator = %s\n", yytext); op++;}
(_?[a-zA-Z])+[a-zA-Z0-9]* {printf("Identifier = %s\n", yytext); id++;}
. {;}
%%

int main()
{
	yylex();
	printf("Keywords = %d\nOperators = %d\nIdentifiers = %d\n", key, op, id);
	return 0;
}
```

OUTPUT:



```
Identifier = a
Identifier = single
Identifier = line
Identifier = comment


Keyword = void
Identifier = fun


Operator = //
Identifier = Inside
Identifier = Fun

Identifier = printf
Identifier = Hello
Identifier = Fun



Operator = //
Identifier = Driver
Identifier = Function

Keyword = int
Identifier = main


Identifier = printf
Identifier = Hello
Identifier = world

Identifier = fun
Operator = //
Identifier = Calling
Identifier = fun

Keyword = return


Keywords = 3
Operators = 6
Identifiers = 28
```

#Contents of sample.c

```
/*
 This is sample.c
*/
#include<stdio.h>

//This is a single line comment.

void fun()
{
	//Inside Fun
	printf("Hello Fun");
}

//Driver Function
int main()
{
	printf("Hello world");
	fun();//Calling fun
	return 0;
}
```
