16.Write a LEX specification count the number of characters, number of lines & number of words.

PROGRAM:

```
%{
int nlines,nwords,nchars;
%}
%%
\n {
    nchars++;nlines++;
    }
[^\n\t]+ {nwords++,nchars=nchars+yyleng;}
. {nchars++;}
%%
int yywrap(void){}
int main()
{
yylex();
printf("Lines=%d\nChars=%d\nWords=%d",nlines,nchars,nwords);
return 0;
}
```

OUTPUT:

```
Command Prompt
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule
"noofcharslineswords.l.txt", line 8: unrecognized rule

C:\Users\bteja\Downloads\cd10>set path=C:\Program Files\GnuWin32\bin

C:\Users\bteja\Downloads\cd10>flex noofcharslineswords.l.txt

C:\Users\bteja\Downloads\cd10>set path=C:\MinGW\bin

C:\Users\bteja\Downloads\cd10>gcc lex.yy.c

C:\Users\bteja\Downloads\cd10>a.exe
bhanuteja will get good package
he will learn required skills
^Z
Lines=2
Chars=62
Words=2
C:\Users\bteja\Downloads\cd10>
```

17. Write a LEX program to print all HTML tags in the input file.

PROGRAM:

```
%{
#include<stdio.h>
%}
%%
\<[^>]*\> fprintf(yyout,"%s/n",yytext);
\n;
%%
int yywrap()
{}
```

```
int main()
{
yyin=fopen("sample.html","r");
yyout=fopen("output.txt","w");
yylex();
}
```

OUTPUT:

```
<html>/n
<body>/n
<h1>/nMy First Heading</h1>/n
<p>/nMy first paragraph.</p>/n
</body>/n
</html>/n
bhanu
teja
```

18.Write a LEX program to count the number of Macros defined and header files included in the C program

PROGRAM:

```
%{
int nmacro,nheader;
%}
%%
^#define {nmacro++;}
^#include {nheader++;}
%%
int yywrap()
{
```

```
return 1;

}

int main()

{

yylex();

printf("NUmber of macros defined=%d\n",nmacro);

printf("NUmber of header files included=%d\n",nheader);

}
```

OUTPUT:

```
Command Prompt

C:\Users\bteja\Downloads>cd p9

C:\Users\bteja\Downloads\p9>set path=C:\Program Files\GnuWin32\bin

C:\Users\bteja\Downloads\p9>flex nmacronheader.l.txt

C:\Users\bteja\Downloads\p9>set path=C:\MinGW\bin

C:\Users\bteja\Downloads\p9>gcc lex.yy.c

C:\Users\bteja\Downloads\p9>a.exe
#include<stdio.h>
<stdio.h>
^Z
NUmber of macros defined=0
NUmber of header files included=1

C:\Users\bteja\Downloads\p9>a.exe
#include<stdio.h>
<stdio.h>
#include<math.h>
<math.h>
#define m=30
 m=30
^Z
NUmber of macros defined=1
NUmber of header files included=2

C:\Users\bteja\Downloads\p9>
```

19. Write a lex program to find whether given input is keyword or identifier?

PROGRAM:

```
%{

%}

%%

if|else|while|int|switch|for|char|double|float|break|continue {printf("it is a keyword");}

[a-zA-Z][a-zA-Z0-9]+ {printf("\n%s is identifier",yytext);}

%%

int yywrap(){}

int main()

{

while(yylex());

}
```

OUTPUT:

```
C:\Users\bteja>cd downloads

C:\Users\bteja\Downloads>cd p10

C:\Users\bteja\Downloads\p10>set path=C:\Program Files\GnuWin32\bin

C:\Users\bteja\Downloads\p10>flex idkey.l.txt

C:\Users\bteja\Downloads\p10>set path=C:\MinGW\bin

C:\Users\bteja\Downloads\p10>gcc lex.yy.c

C:\Users\bteja\Downloads\p10>a.exe
if
it is a keyword
abd

abd is identifier
```

20. Write a lex program for relational operators and words?

PROGRAM:

```
%{
#include <stdio.h>
#include <ctype.h>
%}
%%
[A-Za-z]+      { printf("Word: %s\n", yytext); }
"=="           { printf("Relational Operator: %s\n", yytext); }
"!="           { printf("Relational Operator: %s\n", yytext); }
"<="           { printf("Relational Operator: %s\n", yytext); }
">="           { printf("Relational Operator: %s\n", yytext); }
"<"            { printf("Relational Operator: %s\n", yytext); }
">"            { printf("Relational Operator: %s\n", yytext); }
[ \t\n]+       { /* ignore whitespace */ }
.              { /* ignore other characters */ }
%%
int yywrap() {
    return 1;
}
int main() {
    yylex();
    return 0;
}
```

OUTPUT:

```
C:\Users\bteja\Downloads\p15>set path=C:\Program Files\GnuWin32\bin

C:\Users\bteja\Downloads\p15>flex wordrelop.l.txt

C:\Users\bteja\Downloads\p15>set path=C:\MinGW\bin

C:\Users\bteja\Downloads\p15>gcc lex.yy.c

C:\Users\bteja\Downloads\p15>a.exe
bhanu
Word: bhanu
==
Relational Operator: ==
<=
Relational Operator: <=
>=
Relational Operator: >=
teja
Word: teja
```