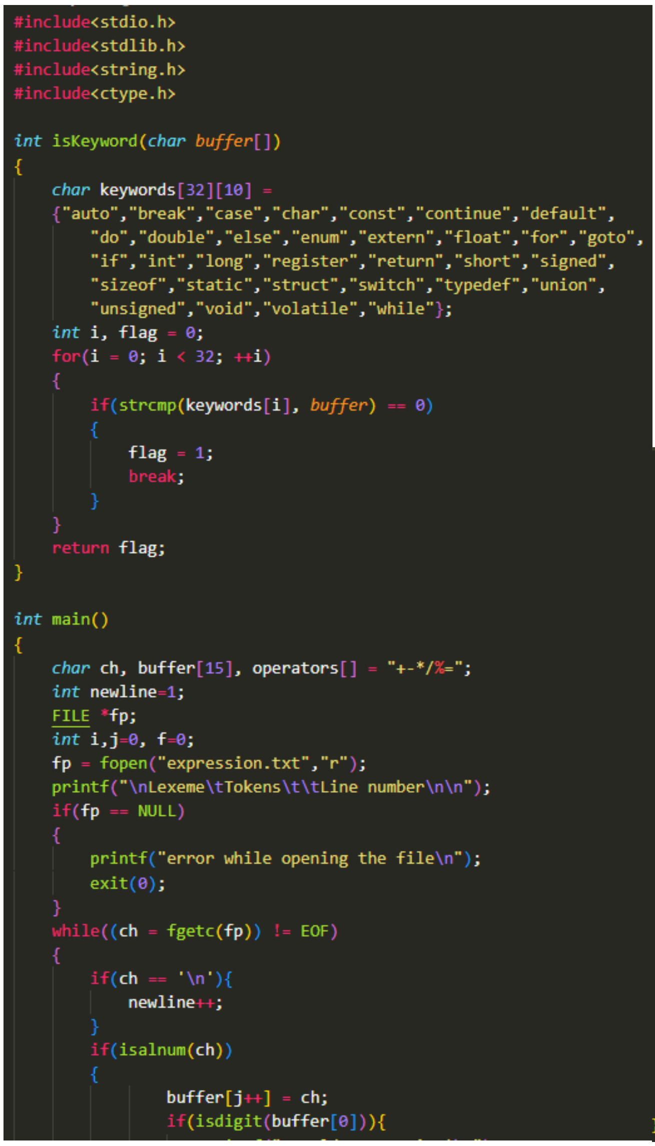
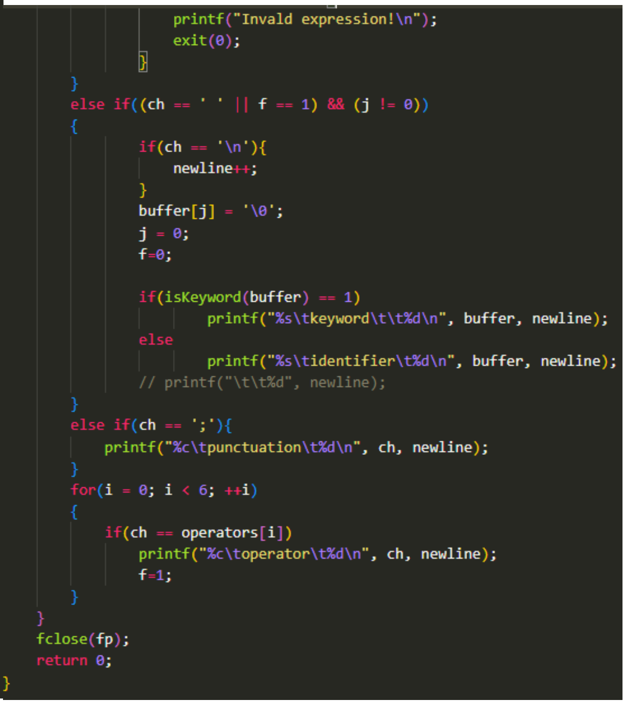
CODES

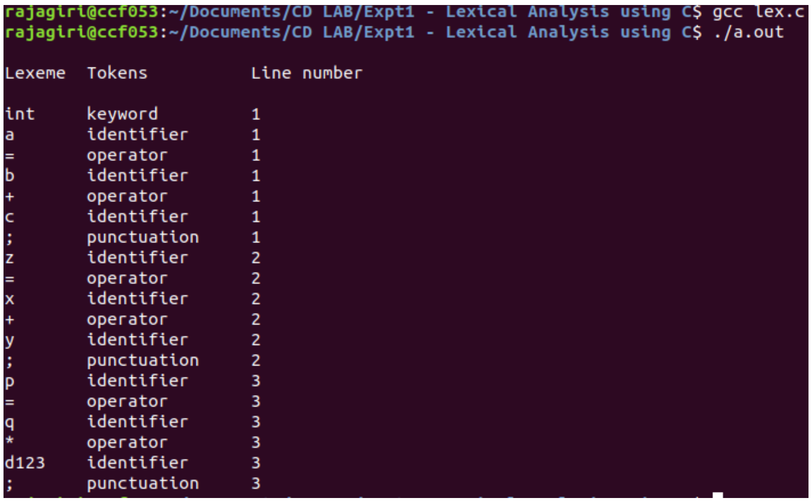


```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>

int isKeyword(char buffer[])
{
    char keywords[32][10] =
    {"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"};
    int i, flag = 0;
    for(i = 0; i < 32; ++i)
    {
        if(strcmp(keywords[i], buffer) == 0)
        {
            flag = 1;
            break;
        }
    }
    return flag;
}

int main()
{
    char ch, buffer[15], operators[] = "+-*/%=";
    int newline=1;
    FILE *fp;
    int i,j=0, f=0;
    fp = fopen("expression.txt","r");
    printf("\nLexeme\tTokens\t\tLine number\n\n");
    if(fp == NULL)
    {
        printf("error while opening the file\n");
        exit(0);
    }
    while((ch = fgetc(fp)) != EOF)
    {
        if(ch == '\n'){
            newline++;
        }
        if(isalnum(ch))
        {
                buffer[j++] = ch;
                if(isdigit(buffer[0])){
```

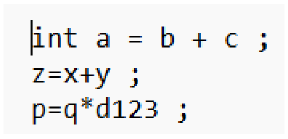


```c
                    printf("Invald expression!\n");
                    exit(0);
                }
        }
        else if((ch == ' ' || f == 1) && (j != 0))
        {
                if(ch == '\n'){
                    newline++;
                }
                buffer[j] = '\0';
                j = 0;
                f=0;

                if(isKeyword(buffer) == 1)
                        printf("%s\tkeyword\t\t%d\n", buffer, newline);
                else
                        printf("%s\tidentifier\t%d\n", buffer, newline);
                // printf("\t\t%d", newline);
        }
        else if(ch == ';'){
            printf("%c\tpunctuation\t%d\n", ch, newline);
        }
        for(i = 0; i < 6; ++i)
        {
            if(ch == operators[i])
                printf("%c\toperator\t%d\n", ch, newline);
                f=1;
        }
    }
    fclose(fp);
    return 0;
}
```

OUTPUTS

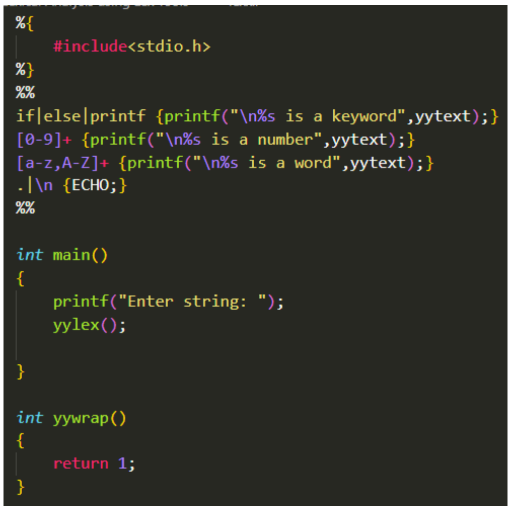


```
rajagiri@ccf053:~/Documents/CD LAB/Expt1 - Lexical Analysis using C$ gcc lex.c
rajagiri@ccf053:~/Documents/CD LAB/Expt1 - Lexical Analysis using C$ ./a.out

Lexeme  Tokens          Line number

int     keyword         1
a       identifier      1
=       operator        1
b       identifier      1
+       operator        1
c       identifier      1
;       punctuation     1
z       identifier      2
=       operator        2
x       identifier      2
+       operator        2
y       identifier      2
;       punctuation     2
p       identifier      3
=       operator        3
q       identifier      3
*       operator        3
d123    identifier      3
;       punctuation     3
```



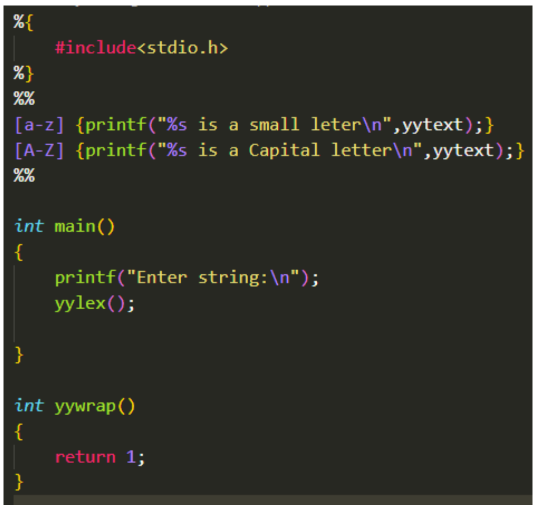
```
int a = b + c ;
z=x+y ;
p=q*d123 ;
```

Keyword



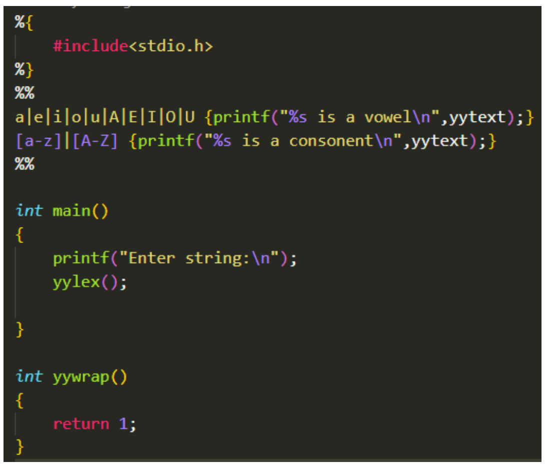
```
%{
    #include<stdio.h>
%}
%%
if|else|printf {printf("\n%s is a keyword",yytext);}
[0-9]+ {printf("\n%s is a number",yytext);}
[a-z,A-Z]+ {printf("\n%s is a word",yytext);}
.|\n {ECHO;}
%%

int main()
{
    printf("Enter string: ");
    yylex();

}

int yywrap()
{
    return 1;
}
```

Uppercase



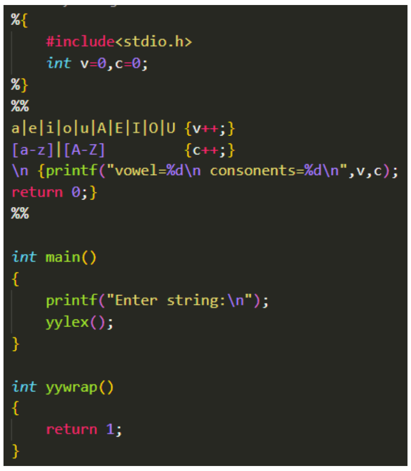
```
%{
    #include<stdio.h>
%}
%%
[a-z] {printf("%s is a small leter\n",yytext);}
[A-Z] {printf("%s is a Capital letter\n",yytext);}
%%

int main()
{
    printf("Enter string:\n");
    yylex();

}

int yywrap()
{
    return 1;
}
```

Vowels



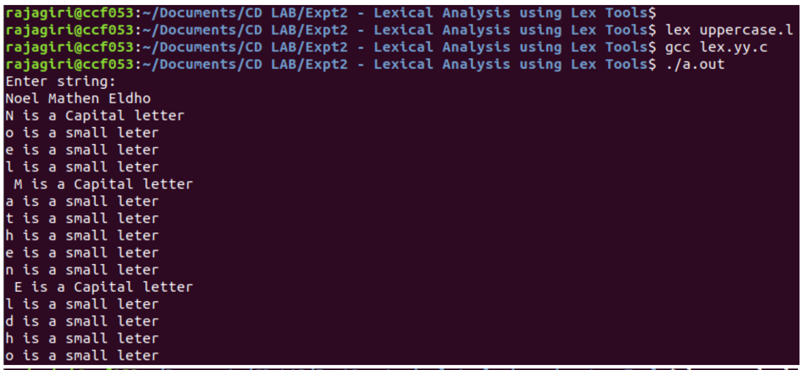
```
%{
    #include<stdio.h>
%}
%%
a|e|i|o|u|A|E|I|O|U {printf("%s is a vowel\n",yytext);}
[a-z]|[A-Z] {printf("%s is a consonent\n",yytext);}
%%

int main()
{
    printf("Enter string:\n");
    yylex();

}

int yywrap()
{
    return 1;
}
```

Count

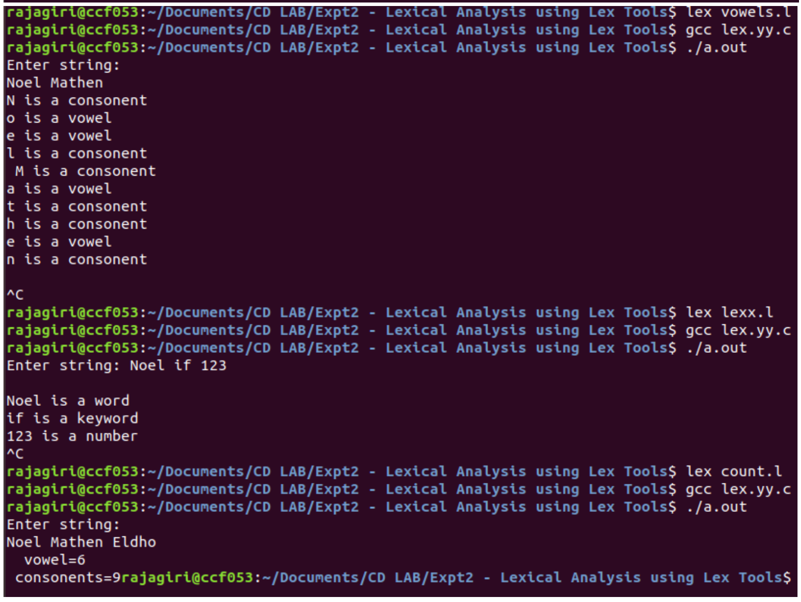


```
%{
    #include<stdio.h>
    int v=0,c=0;
%}
%%
a|e|i|o|u|A|E|I|O|U {v++;}
[a-z]|[A-Z]         {c++;}
\n {printf("vowel=%d\n consonents=%d\n",v,c);
return 0;}
%%

int main()
{
    printf("Enter string:\n");
    yylex();
}

int yywrap()
{
    return 1;
}
```

OUTPUTS



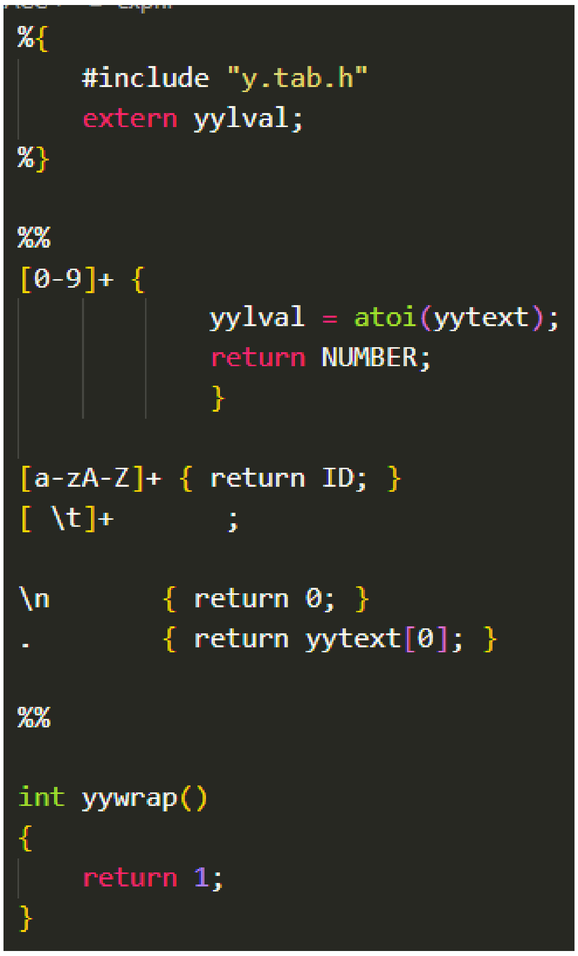
```
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ lex uppercase.l
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ gcc lex.yy.c
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ ./a.out
Enter string:
Noel Mathen Eldho
N is a Capital letter
o is a small leter
e is a small leter
l is a small leter
 M is a Capital letter
a is a small leter
t is a small leter
h is a small leter
e is a small leter
n is a small leter
 E is a Capital letter
l is a small leter
d is a small leter
h is a small leter
o is a small leter
```



```
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ lex vowels.l
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ gcc lex.yy.c
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ ./a.out
Enter string:
Noel Mathen
N is a consonent
o is a vowel
e is a vowel
l is a consonent
 M is a consonent
a is a vowel
t is a consonent
h is a consonent
e is a vowel
n is a consonent

^C
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ lex lexx.l
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ gcc lex.yy.c
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ ./a.out
Enter string: Noel if 123

Noel is a word
if is a keyword
123 is a number
^C
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ lex count.l
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ gcc lex.yy.c
rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$ ./a.out
Enter string:
Noel Mathen Eldho
  vowel=6
 consonents=9rajagiri@ccf053:~/Documents/CD LAB/Expt2 - Lexical Analysis using Lex Tools$
```

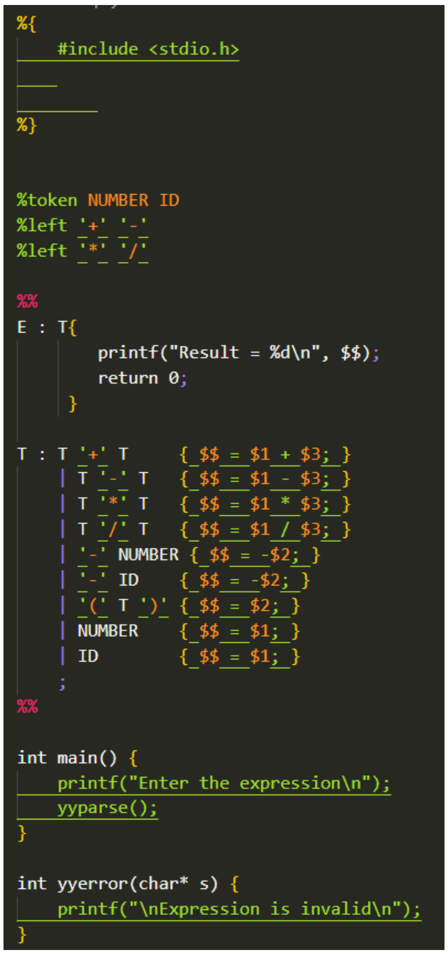
CODES

.l file



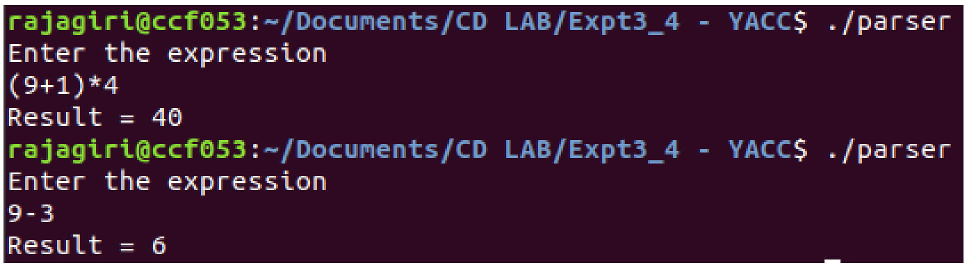
```
%{
    #include "y.tab.h"
    extern yylval;
%}

%%
[0-9]+ {
                yylval = atoi(yytext);
                return NUMBER;
                }

[a-zA-Z]+ { return ID; }
[ \t]+        ;

\n        { return 0; }
.         { return yytext[0]; }

%%

int yywrap()
{
    return 1;
}
```

.y file



```
%{
    #include <stdio.h>


%}


%token NUMBER ID
%left '+' '-'
%left '*' '/'

%%
E : T{
        printf("Result = %d\n", $$);
        return 0;
     }

T : T '+' T     { $$ = $1 + $3; }
    | T '-' T   { $$ = $1 - $3; }
    | T '*' T   { $$ = $1 * $3; }
    | T '/' T   { $$ = $1 / $3; }
    | '-' NUMBER { $$ = -$2; }
    | '-' ID    { $$ = -$2; }
    | '(' T ')' { $$ = $2; }
    | NUMBER    { $$ = $1; }
    | ID        { $$ = $1; }
    ;
%%

int main() {
    printf("Enter the expression\n");
    yyparse();
}

int yyerror(char* s) {
    printf("\nExpression is invalid\n");
}
```

OUTPUTS



```
rajagiri@ccf053:~/Documents/CD LAB/Expt3_4 - YACC$ ./parser
Enter the expression
(9+1)*4
Result = 40
rajagiri@ccf053:~/Documents/CD LAB/Expt3_4 - YACC$ ./parser
Enter the expression
9-3
Result = 6
```