LEX:

1a. Program to count the number of characters, words, spaces and lines in a given input file.

Program:

```
%{
int ch=0,wd=0,ln=0,sp=0;
%}
%%
" " {sp++;wd++;}
[\n] {ln++;wd++;}
[\t\n] {wd++;}
[^\t\n] {ch++;}
%%
int yywrap()
{
return 1;
}
int main()
{
yyin=fopen("a.txt","r");
yylex();
printf("char=%d\t words=%d\t spaces=%d\t lines=%d",ch,wd,sp,ln);
return 0;
}
```

Create a text file named a.txt:

This file contains:

Compiler Design
System Software

```
[vizion@localhost ~]$ cat > a.txt
Compiler Design
System Software
```

Compile and Run the program:

```
[vizion@localhost ~]$ lex lines.l
[vizion@localhost ~]$ cc lex.yy.c -ll
[vizion@localhost ~]$ ./a.out
char=28  words=4        spaces=2        lines=2
```

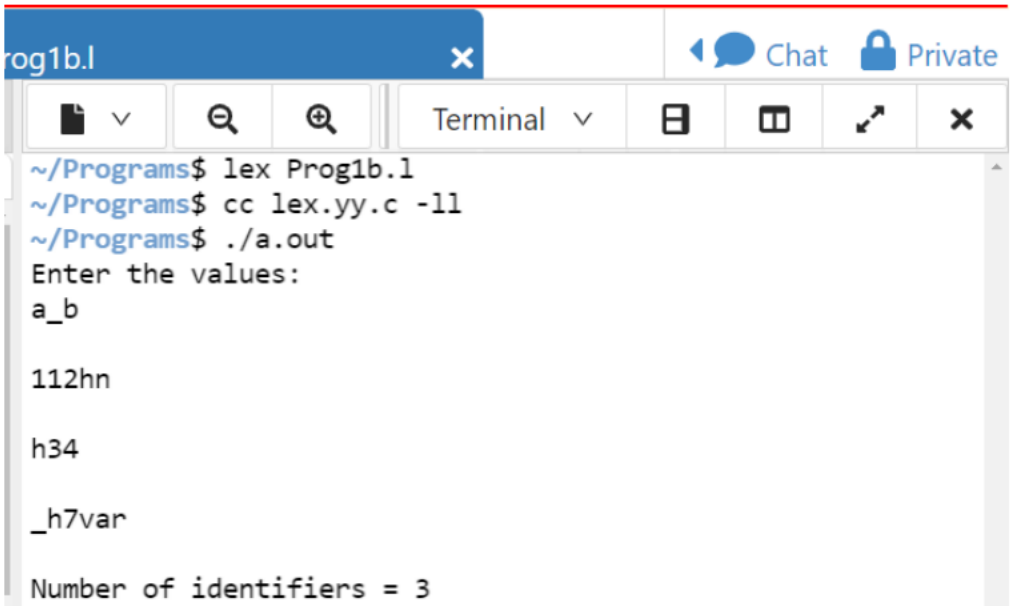
1b. Program to recognize and count the number of identifiers in a file

Code:

```
%{
#include<stdio.h>
int i=0;
%}
digit [0-9]
letter [a-z A-Z_]
%%
{letter}({letter}|{digit})* {i++;}
{digit}({letter}|{digit})* {i;}
%%
int main()
{
printf("Enter the values:\n");

yylex();
printf("Number of identifiers = %d\n", i);
return 0;
}
```

Output:



2a. Program to count the numbers of comment lines in a given C program.Also eliminate them and copy the resulting program into separate file.

```
%{
#include<stdio.h>
int ml=0;
int sl=0;
%}
%%
"/*"[a-zA-Z0-9' '\t\n]*"*/" ml++;
"//".*  sl++;
%%
 main()
{
yyin=fopen("f1.txt","r");
yyout=fopen("f2.txt","w");
yylex();
fclose(yyin);
fclose(yyout);
printf("Number of single line comments=%d\n",sl);
printf("Number of multiline comments=%d\n",ml);
}
```

Output:

Create a file f1.txt with comment lines

```
[vizion@localhost ~]$ cat f1.txt
/*Definition of compiler*/
Compiler is a program that converts code written
in one programming language to other
//definition of assmebler
Assembler converts symbolic code to machine code
/*end of file*/
```

Run the program

```
[vizion@localhost ~]$ lex 2a.l
[vizion@localhost ~]$ cc lex.yy.c -ll
[vizion@localhost ~]$ ./a.out f1.txt f2.txt
Number of single line comments=1
Number of multiline comments=2
```

Display the contents of file f2.txt . Comment lines are eliminated:

```
[vizion@localhost ~]$ cat f2.txt

Compiler is a program that converts code written
in one programming language to other

Assembler converts symbolic code to machine code
```

2b. Program to recognize whether a given sentence is simple or compound.

```
%{
#include<stdio.h>
int valid;
%}
%%
[a-zA-Z][ ](and|but|or|however)[ ][a-zA-Z]* {valid=1;}
.|[\n];
%%
int main()
{
printf("enter the text\n");
yylex();
if(valid)
{
printf("\n statement is compound\n");
}
else
{
printf("\n statement is simple\n");
}
return 0;
}
```

Output:

```
[vizion@localhost ~]$ lex 2b.l
[vizion@localhost ~]$ cc lex.yy.c -ll
[vizion@localhost ~]$ ./a.out
enter the text
Compiler Design and system software


 statement is compound
[vizion@localhost ~]$
[vizion@localhost ~]$ ./a.out
enter the text
loader is a part of OS


 statement is simple
```

3a. Program to count no of:

i.+ve and –ve integers

ii. +ve and –ve fractions

```
%{
    #include <stdio.h>
    int pi=0,ni=0,pf=0,nf=0;
%}
%%
[-][0-9]+ {ni++;}
[+]?[0-9]+ {pi++;}
[-][0-9]*\.[0-9]+ {nf++;}
[+]?[0-9]*\.[0-9]+ {pf++;}
%%

void main(int argc,char *argv[])
{
if (argc!=2)
{
 printf("usage : ./a.out in.txt \n");
 exit(0);
}
yyin=fopen(argv[1],"r");
yylex();
printf("no. of positive integer %d \n",pi);
printf("no. of negative integer %d \n",ni);
printf("no. of positive fraction %d \n",pf);
printf("no. of negative fraction %d \n",nf);
}
 int yywrap()
{
return 1;
}
```

3b. Program to count the no of 'scan f' and 'printf' statements in a C program. Replace them with 'readf' and 'writef' statements respectively.

```
%{
#include<stdio.h>
int sf=0;pf=0;
%}
%%
"scanf" {sf++; fprintf(yyout,"readf");}
"printf" {pf++; fprintf(yyout,"writef");}
%%

int main()
{
yyin=fopen("file1.c","r");
yyout=fopen("file2.c","w");
yylex();
printf("Number of scanf=%d\n Number of printf=%d\n",sf,pf);
return 0;
}
```

Output:

Input file-file1.c which has printf and scanf statements

```
[vizion@localhost ~]$ cat file1.c
#include<stdio.h>
int main()
{
int a,b,c;
printf("Enter the values of a and b\n");
scanf("%d %d",&a,&b);
c=a+b;
printf("Sum=%d",c);
return 0;
}
```

Run the program and display the contents of file2.c in which printf and scanf are replaced by writef and readf

```
[vizion@localhost ~]$ vi 3b.l
[vizion@localhost ~]$ lex 3b.l
[vizion@localhost ~]$ cc lex.yy.c -ll
3b.l:3: warning: data definition has no type or storage class
[vizion@localhost ~]$ ./a.out
Number of scanf=1
 Number of printf=2
[vizion@localhost ~]$ cat file2.c
#include<stdio.h>
int main()
{
int a,b,c;
writef("Enter the values of a and b\n");
readf("%d %d",&a,&b);
c=a+b;
writef("Sum=%d",c);
return 0;
}
```

YACC:

4. Program to evaluate arithmetic expression involving operators +,-,*,/

Lex Part

```
%{
#include "y.tab.h"
extern yylval;
%}
%%
[0-9]+ {yylval=atoi(yytext);return num;}        /* convert the string to
                                                   number and send the
                                                   value*/

[\+\-\*\/] {return yytext[0];}
[)] {return yytext[0];}
[(] {return yytext[0];}
. {;}
\n {return 0;}
%%
```

YACC Part

```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token num
%left '+' '-'
%left '*' '/'
%%
input:exp {printf("%d\n",$$);exit(0);}
exp:exp'+'exp {$$=$1+$3;}
 |exp'-'exp{$$=$1-$3;}
 |exp'*'exp{$$=$1*$3;}

|exp'/'exp { if($3==0){printf("Divide by Zero error\n");exit(0);}
            else
              $$=$1/$3;}
|'('exp')'{$$=$2;}
|num{$$=$1;};
%%
int yyerror()
{
        printf("error");
        exit(0);
}
int main()
{
        printf("Enter an expression:\n");
        yyparse();
}
```

Output:

```
[vizion@localhost ~]$ lex 4a.l
[vizion@localhost ~]$ yacc -d 4a.y
[vizion@localhost ~]$ cc lex.yy.c y.tab.c -lfl
[vizion@localhost ~]$ ./a.out
enter an expression
8/4+6-1
7
```

5. **Program to recognize a valid variable which starts with a letter, followed by any number of letters or digits.**

Lex part

```
%{
#include "y.tab.h"
%}
%%
[a-z] return L;
[0-9] return D;

%%
```

Yacc part

```
%{
%}
%token L D
%%
var : L E  { printf(" Valid Variable \n"); return 0;}
E: E L     ;
|E D       ;
|          ;
%%
main()
        { printf(" Type the Variable \n");   yyparse();
        } yyerror()
        {  printf(" Invalid variable !!!\n");  exit(0); }
```

Sample Input/Output:

```
$lex 4b.l

$yacc –d 4b.y

$cc lex.yy.c  y.tab.c  -lfl
$./a.out
Sum6
The string is a valid variable
$./a.out
4Sum
 The string is not a valid variable
```

6.Program to recognize the strings using the grammar ($a^n b^n$;n>=0)

```
%{
#include "y.tab.h"
%}
%%
a return A;
b return B;
.  return yytext[0];
\n return yytext[0];
%%
```

```
%{
#include<stdio.h>
%}
%token A B
%%
str:s '\n' { return 0; }
s:A s B;
| ;
%%
main()
{
printf("Type the string ?\n");
 if(!yyparse())
printf("Valid string");
}
int yyerror()
{
printf("invalid string");
exit(0);
}
```

Output

```
[vizion@localhost ~]$ lex 6a.l
[vizion@localhost ~]$ yacc 6a.y
[vizion@localhost ~]$ cc lex.yy.c y.tab.c -lfl
[vizion@localhost ~]$ ./a.out
Type the string ?
aabb
Valid string[vizion@localhost ~]$ ./a.out
Type the string ?
aabbb
```