System Programming Practical

M D Kaif(20020570019)
BSc Computer Science (Hons)

Practical

1. Write a Lex program to count the number of lines and characters in the input file.

Lex file

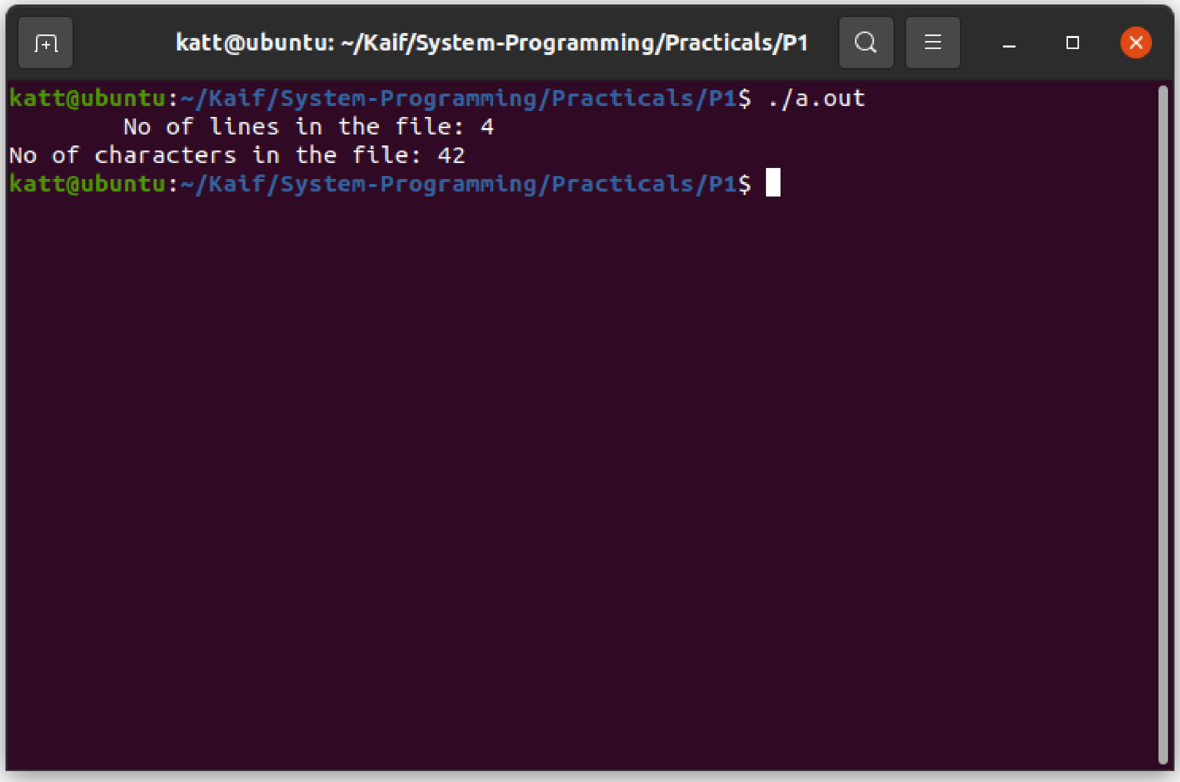
```
%{
#include<stdio.h>
int lines=0, chars=0;
%}
%%
[\n] {lines++;}
[a-zA-Z0-9] {chars++;}
[^ \t \n]+ {chars+=yyleng;} //yyleng= length of the matched string
%%
int main()
{
yyin= fopen("input.txt", "r");
yylex();
printf("No of lines in the file: %d", lines);
printf("\n");
printf("No of characters in the file: %d", chars);
printf("\n");
return 0;
}
int yywrap()
{
return 1;
}
```

Input.txt

```
Hello World
```

Proof lies in sorce code

Good To see You



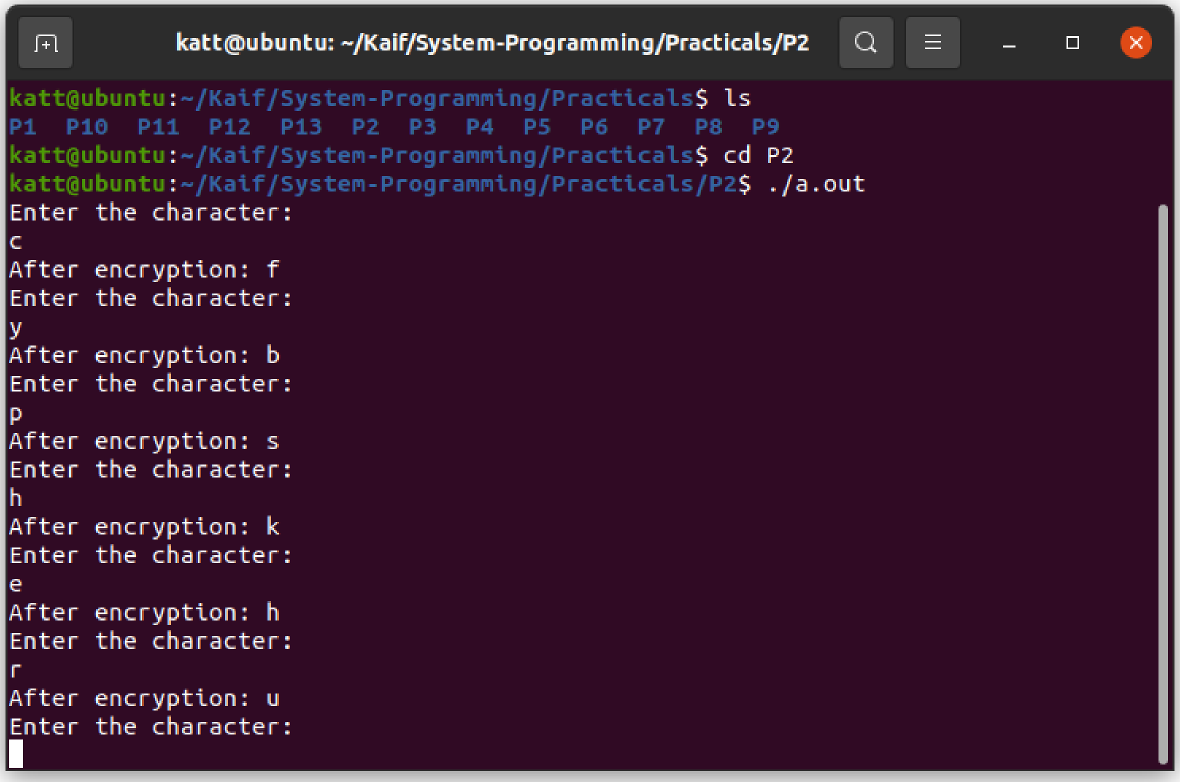
```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P1$ ./a.out
        No of lines in the file: 4
No of characters in the file: 42
katt@ubuntu:~/Kaif/System-Programming/Practicals/P1$ 
```

2. Write a Lex program that implements the Caesar cipher: it replaces every letter with the one three letters after in alphabetical order, wrapping around at Z. e.g. a is replaced by d, b by e,
and so on z by c.

Lex file

```
%{
#include <stdio.h>
%}
%%
[a-z] {char ch=yytext[0];
ch+=3;
if(ch>'z')
ch-=('z'+1-'a');
printf("After encryption: %c", ch);printf("\nEnter the
character: ");
}
[A-Z] {char ch=yytext[0];
ch+=3;
if(ch>'Z')
ch-=('Z'+1-'A');
printf("After encryption: %c", ch);
printf("\nEnter the character: ");
}
%%
int yywrap()
```

```
{
return 1;
}

int main()
{
printf("Enter the character: \n");
yylex();
return 0;
}
```



katt@ubuntu: ~/Kaif/System-Programming/Practicals/P2

```
katt@ubuntu:~/Kaif/System-Programming/Practicals$ ls
P1  P10  P11  P12  P13  P2  P3  P4  P5  P6  P7  P8  P9
katt@ubuntu:~/Kaif/System-Programming/Practicals$ cd P2
katt@ubuntu:~/Kaif/System-Programming/Practicals/P2$ ./a.out
Enter the character:
c
After encryption: f
Enter the character:
y
After encryption: b
Enter the character:
p
After encryption: s
Enter the character:
h
After encryption: k
Enter the character:
e
After encryption: h
Enter the character:
r
After encryption: u
Enter the character:
```

3. Write a Lex program that finds the longest word (defined as a contiguous string of upper and lower-case letters) in the input.

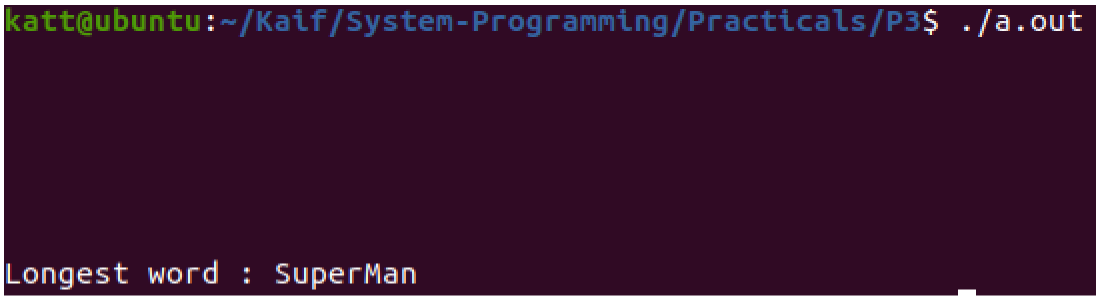
Lex file

```
%{
#include<stdio.h>
#include<strings.h>
// initialising length
int length=0;
// char array for storing longest word
char longestword[50];
%}

%%
[A-Za-z0-9]+ { if (yyleng > length) {

        length=yyleng;
       // strcpy function to copy current word in yytxt in
longest
        strcpy(longestword,yytext);
        }
    }
"." return 1;
%%
int main()
{
```

```
yyin=fopen("input.txt","r");

yylex();
printf("Longest word : %s\n",longestword);
//printf("Length of Longest word : %s\n",length);

return 0;
}
int yywrap(){
	return 1;
}
```

input.txt

```
Hello World

Proof lies in source code
Good To see You

I am SuperMan
```



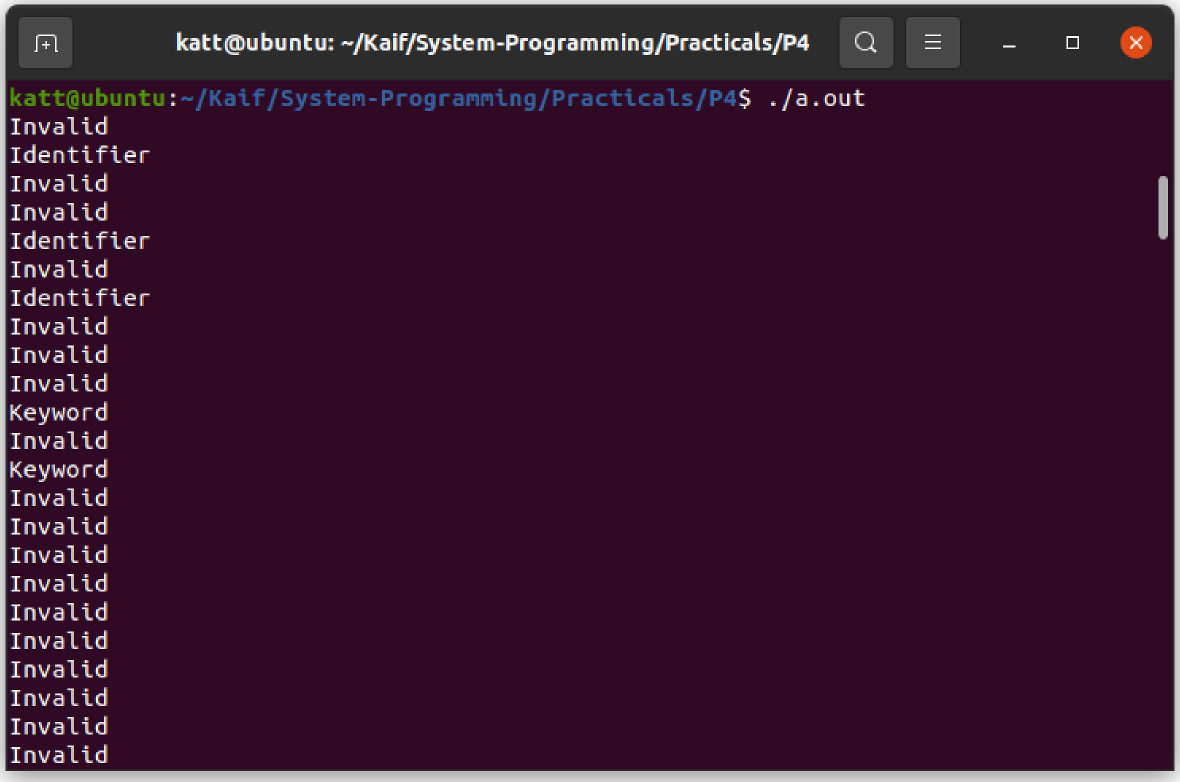
```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P3$ ./a.out




Longest word : SuperMan
```

4. Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators, and comments in any simple programming language.

Lex file

```
%{

%}
%%
[0-9]* {printf("Integer\n");}
[0-9]+\.[0-9]+ {printf("Float\n"); }
int|float|if|else|printf|main|exit|switch {printf("Keyword\n");}
[+|*|/|%|&] {printf("Operators\n");}
"-" {printf("Operators\n");}
"/*".*"*/" {printf("comment\n");}
[_a-zA-Z][_a-zA-Z0-9]{0,30} {printf("Identifier\n");}
. {printf("Invalid\n");}
%%
int main()
{
yyin=fopen("code.c","r");
yyout=fopen("input.txt","w");
yylex();

}
int yywrap()
{
```

```
return 1;
}

C file
#include <stdio.h>
int main()
{
        int num, i; // declare a variable
        printf (" Enter a number to generate the table in C: ");
        scanf (" %d", &num); // take a positive number from
the user

        printf ("\n Table of %d", num);
        // use for loop to iterate the number from 1 to 10
        for ( i = 1; i <= 10; i++)
        {
        printf ("\n %d * %d = %d", num, i, (num*i));
        }
        return 0;
}
```



```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P4$ ./a.out
Invalid
Identifier
Invalid
Invalid
Identifier
Invalid
Identifier
Invalid
Invalid
Invalid
Keyword
Invalid
Keyword
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
Invalid
```

5. Write a Lex program to count the number of identifiers in a C file.

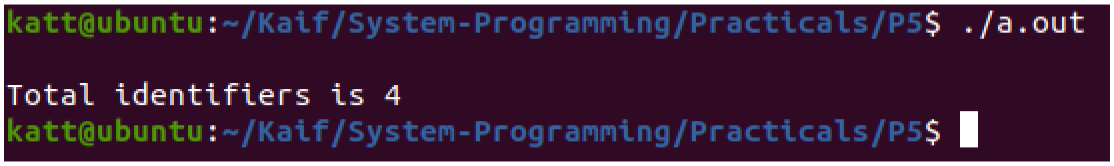
Lex file

```
%{
char ch;
int id;
%}
%%
^[ \t]*(int|float|double|char) {
ch=input();
while(1)
{
if(ch==',')
id++;else if(ch==';')
{
id++;
break;
}
ch=input();
}
}
.|[\n] ;
%%
int yywrap(){
return 1;
}
```

```
int main()
{
yyin=fopen("input.c","r");
yylex();
printf("\nTotal identifiers is %d\n",id);
}
```

C file

```
#include<iostream>
#include<cstdio>
int main()
{
int num,a,b,c;
printf("Enter a number: ");
scanf("%d",&num);
printf("You have entered %d",num);
return 0;
}
```



```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P5$ ./a.out

Total identifiers is 4
katt@ubuntu:~/Kaif/System-Programming/Practicals/P5$
```

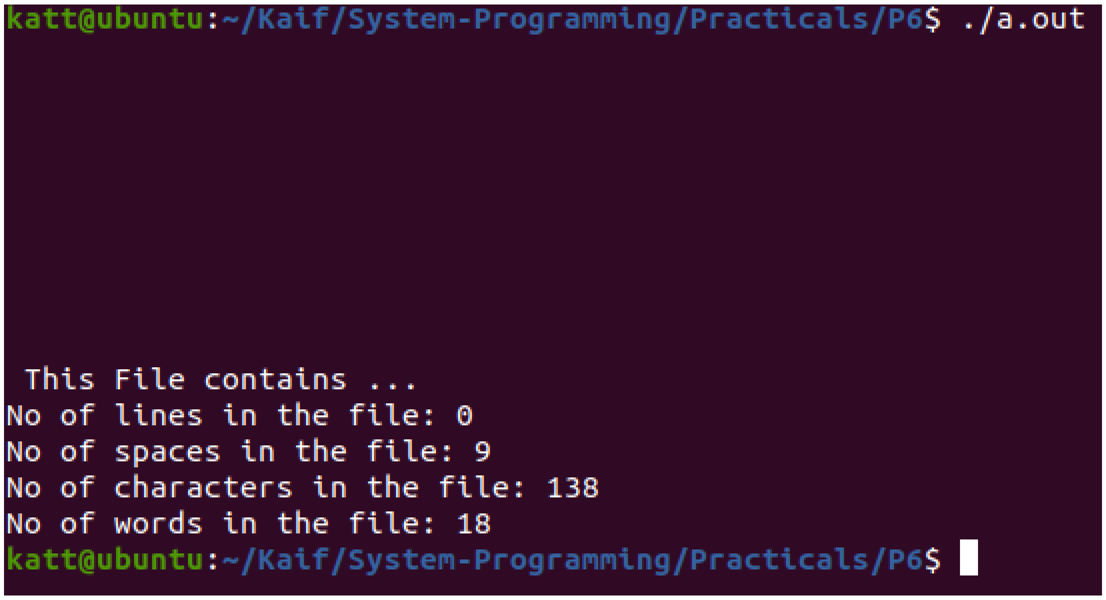
6. Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.

Lex file

```
%{
#include<stdio.h>
int lines =0, chars= 0, spaces=0, words=0;
%}
%%[\n] {lines++;}
[ ]|[\t] {spaces++;}
[^ \t \n]+ {words++; chars+=yyleng;}
%%
int main()
{
yyin= fopen("input.c", "r");
yylex();
printf(" This File contains ...");
printf("\n");
printf("No of lines in the file: %d", lines);
printf("\n");
printf("No of spaces in the file: %d", spaces);
printf("\n");
printf("No of characters in the file: %d", chars);
printf("\n");
printf("No of words in the file: %d", words);
printf("\n");
return 0;
}
int yywrap()
{return 1;}
```

C file

```
#include<iostream>
#include<cstdio>
int main()
{
int num,a,b,c;
printf("Enter a number: ");
scanf("%d",&num);
printf("You have entered %d",num);
return 0;}
```



```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P6$ ./a.out








 This File contains ...
No of lines in the file: 0
No of spaces in the file: 9
No of characters in the file: 138
No of words in the file: 18
katt@ubuntu:~/Kaif/System-Programming/Practicals/P6$
```

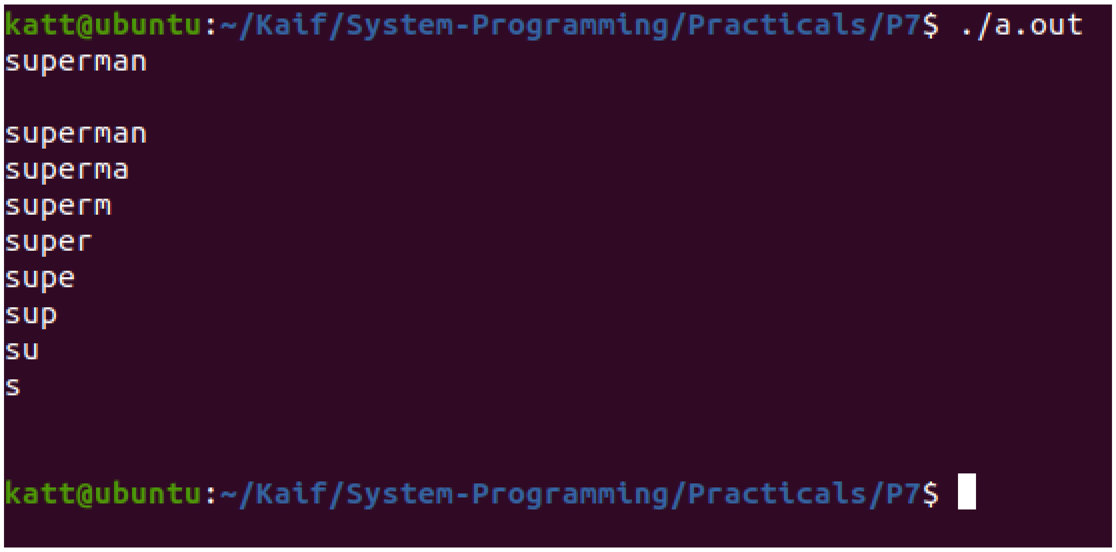
7. Write a Lex specification program that generates a C program which takes a string “abcd”
and prints the following output.

```
abcd
abc
ab
a
```

Lex file

```
%{
#include<stdio.h>
#include<string.h>
char ch[8];
int i,j;
%}
char [a-zA-Z]
%%
{char}+ {
printf("\n");
for(i=yyleng;i>=0;i--)
{
for( j=0;j<i;j++)
printf("%c",yytext[ j]);
printf("\n");
}
}
%%
```

```
int yywrap()
{
return 1;
}
int main()
{
yylex();
return 0;
}
```



```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P7$ ./a.out
superman

superman
superma
superm
super
supe
sup
su
s


katt@ubuntu:~/Kaif/System-Programming/Practicals/P7$
```

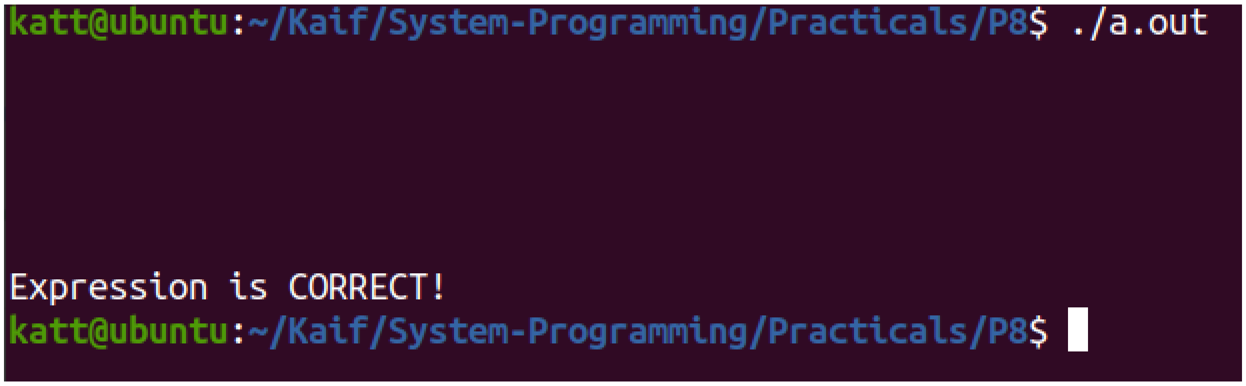
8. A program in Lex to recognize a valid arithmetic expression.

Lex file

```
%{
#include<strings.h>
int opcount=0,intcount=0,check=1,top=0;
%}
%%
['('] {check=0;}
[')'] {check=1;}
[+|*|/|-] {opcount++;}
[0-9]+ {intcount++;}
. {printf("Invalid Input only digits and +|-|*|/ is valid\n");}
%%
int main()
{

yyin=fopen("input.txt","r");
yylex();
if(intcount=opcount+1)
{
 if(check==1)
 {
       printf("Expression is CORRECT!\n");
 }
 else{
        printf("')' bracket missing from expression\n");
 }
}
else{
   printf("Expression is INCORRECT!\n");
}
}

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

input.txt

```
45+23
49*66+63
*562/5
7*8*856+23
```



```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P8$ ./a.out




Expression is CORRECT!
katt@ubuntu:~/Kaif/System-Programming/Practicals/P8$ 
```

9. Write a YACC program to find the validity of a given expression (for operators + - * and /)

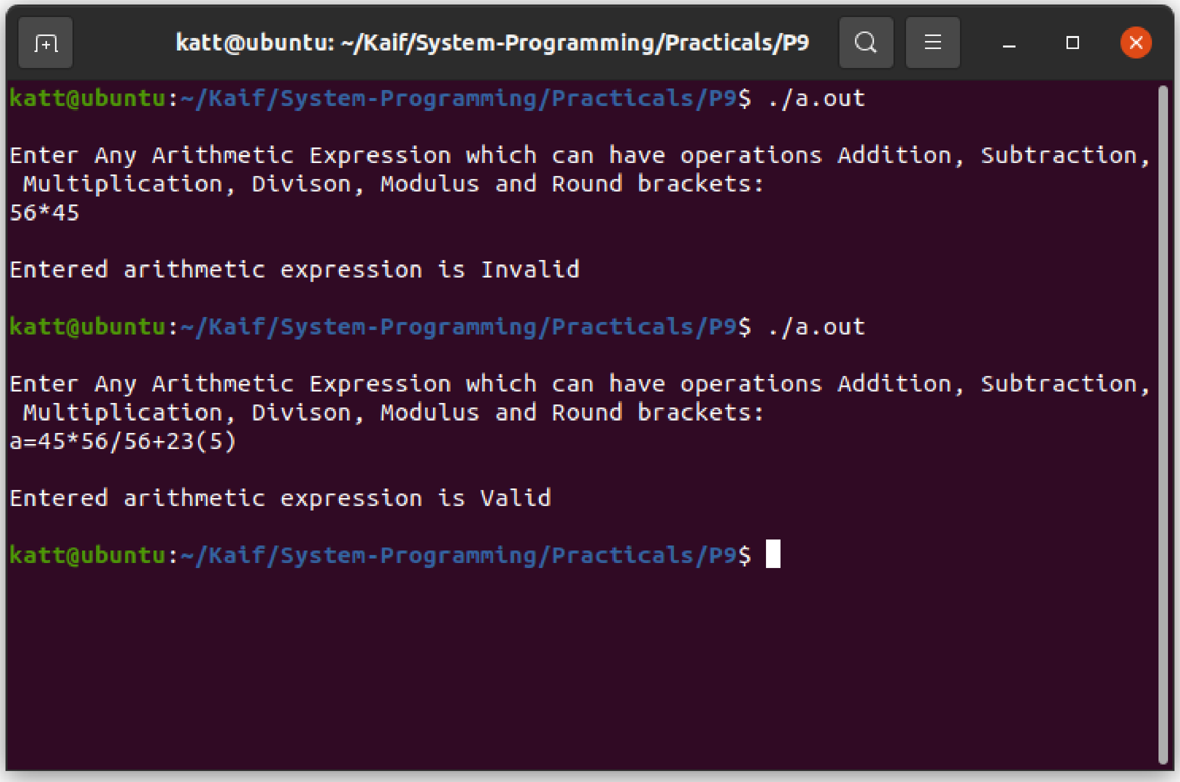
Lex file

```
%{
#include<stdio.h>
#include "y.tab.h"
%}

%%
[a-zA-Z]+ return VARIABLE;
[0-9]+ return NUMBER;
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

Yacc file

```
%{
      #include<stdio.h>
%}
%token NUMBER
%token VARIABLE
```

```
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'

%%

S: VARIABLE'='E {
        printf("\nEntered arithmetic expression is Valid\n\n");
        return 0;
        }
E:E'+'E
 |E'-'E
 |E'*'E
 |E'/'E
 |E'%'E
 |'('E')'
 | NUMBER
 | VARIABLE
;

%%

void main()
{
    printf("\nEnter Any Arithmetic Expression which can
have operations Addition, Subtraction, Multiplication,
Divison, Modulus and Round brackets:\n");
    yyparse();
}

void yyerror()
{
    printf("\nEntered arithmetic expression is Invalid\n\n");

}
```



```
katt@ubuntu: ~/Kaif/System-Programming/Practicals/P9
katt@ubuntu:~/Kaif/System-Programming/Practicals/P9$ ./a.out

Enter Any Arithmetic Expression which can have operations Addition, Subtraction,
 Multiplication, Divison, Modulus and Round brackets:
56*45

Entered arithmetic expression is Invalid

katt@ubuntu:~/Kaif/System-Programming/Practicals/P9$ ./a.out

Enter Any Arithmetic Expression which can have operations Addition, Subtraction,
 Multiplication, Divison, Modulus and Round brackets:
a=45*56/56+23(5)

Entered arithmetic expression is Valid

katt@ubuntu:~/Kaif/System-Programming/Practicals/P9$
```

10. A Program in YACC which recognizes a valid variable which starts with letter followed by a digit. The letter should be in lowercase only.

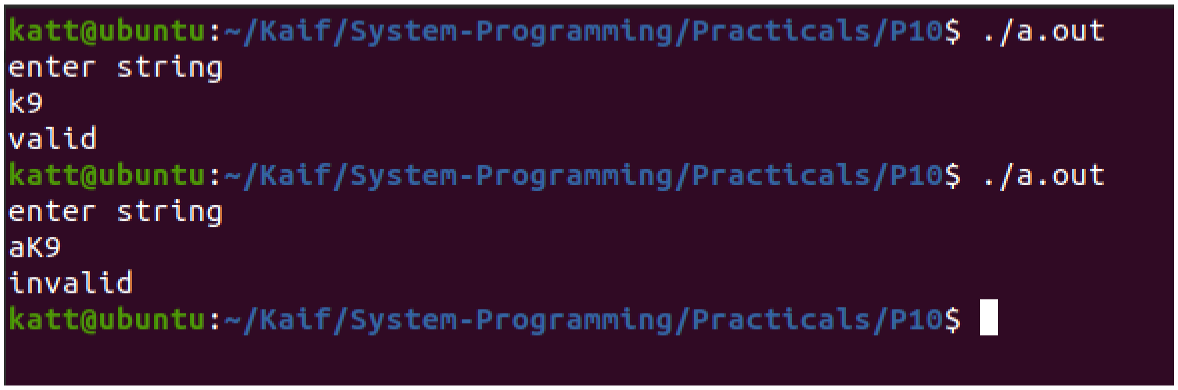
Lex file

```
%{
#include "y.tab.h"
%}
%%
[0-9]+ {return DIGIT;}
[a-z]+ {return LETTER;}
[ \t] {;}
\n { return 0;}
. {return yytext[0];}
%%
```

Yacc file

```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token DIGIT LETTER
%%
stmt:A
       ;
A: LETTER B
 ;
```

```
B: LETTER B
 | DIGIT B
 | LETTER
 | DIGIT
 ;
%%
void main(){
printf("enter string \n");
yyparse();
printf("valid \n");
exit(0);
}
void yyerror()
{
printf("invalid \n");
exit(0);
}
```



```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P10$ ./a.out
enter string
k9
valid
katt@ubuntu:~/Kaif/System-Programming/Practicals/P10$ ./a.out
enter string
aK9
invalid
katt@ubuntu:~/Kaif/System-Programming/Practicals/P10$
```

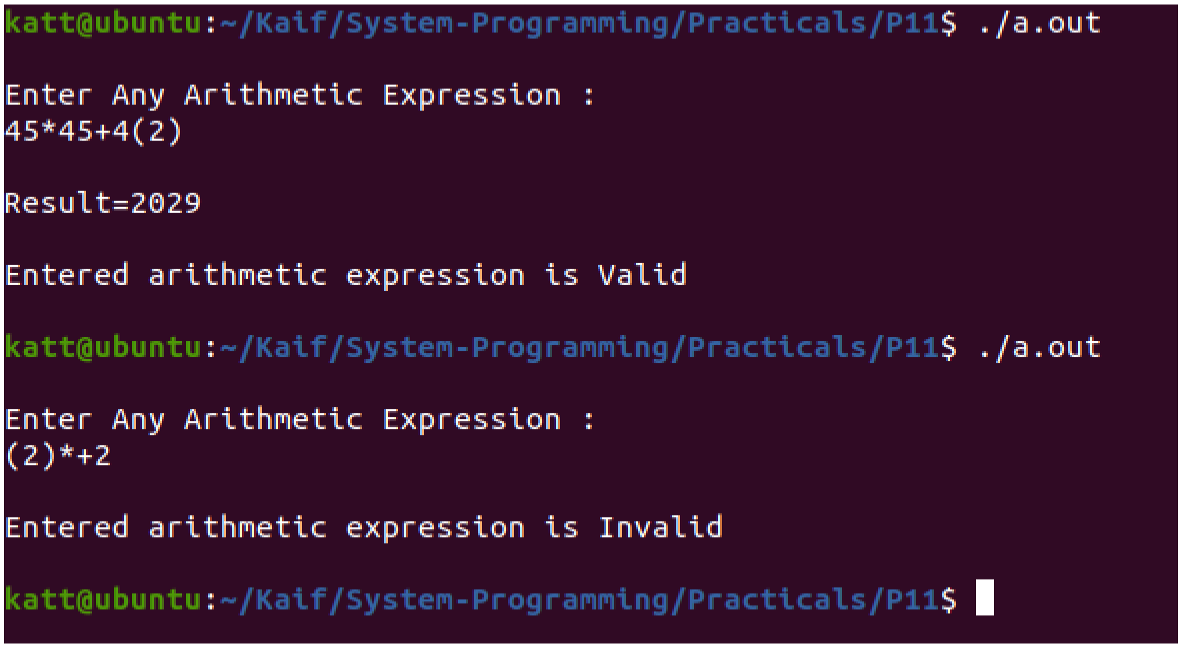
11. A Program in YACC to evaluate an expression (simple calculator program for addition and subtraction, multiplication, division).

Lex file

```
%{
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
%}

%%
[0-9]+ {
       yylval=atoi(yytext);
       return NUMBER;
       }
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

Yacc file

```
%{
       #include<stdio.h>
       int flag=0;

%}
%token NUMBER

%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E{
       printf("\nResult=%d\n",$$);
       return 0;
       }
E:E'+'E {$$=$1+$3;}
 |E'-'E {$$=$1-$3;}
 |E'*'E {$$=$1*$3;}
 |E'/'E {$$=$1/$3;}
 |E'%'E {$$=$1%$3;}
 |'('E')' {$$=$2;}
 | NUMBER {$$=$1;}
;
%%

void main()
{
   printf("\nEnter Any Arithmetic Expression :\n");
   yyparse();
  if(flag==0)
   printf("\nEntered arithmetic expression is Valid\n\n");

}
void yyerror()
{
   printf("\nEntered arithmetic expression is Invalid\n\n");
   flag=1;
}
```



```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P11$ ./a.out

Enter Any Arithmetic Expression :
45*45+4(2)

Result=2029

Entered arithmetic expression is Valid

katt@ubuntu:~/Kaif/System-Programming/Practicals/P11$ ./a.out

Enter Any Arithmetic Expression :
(2)*+2

Entered arithmetic expression is Invalid

katt@ubuntu:~/Kaif/System-Programming/Practicals/P11$
```

12. Program in YACC to recognize the strings “ab”, “aabb”, ”aaabbb”,... of the language ($an\ bn$,n>=1).

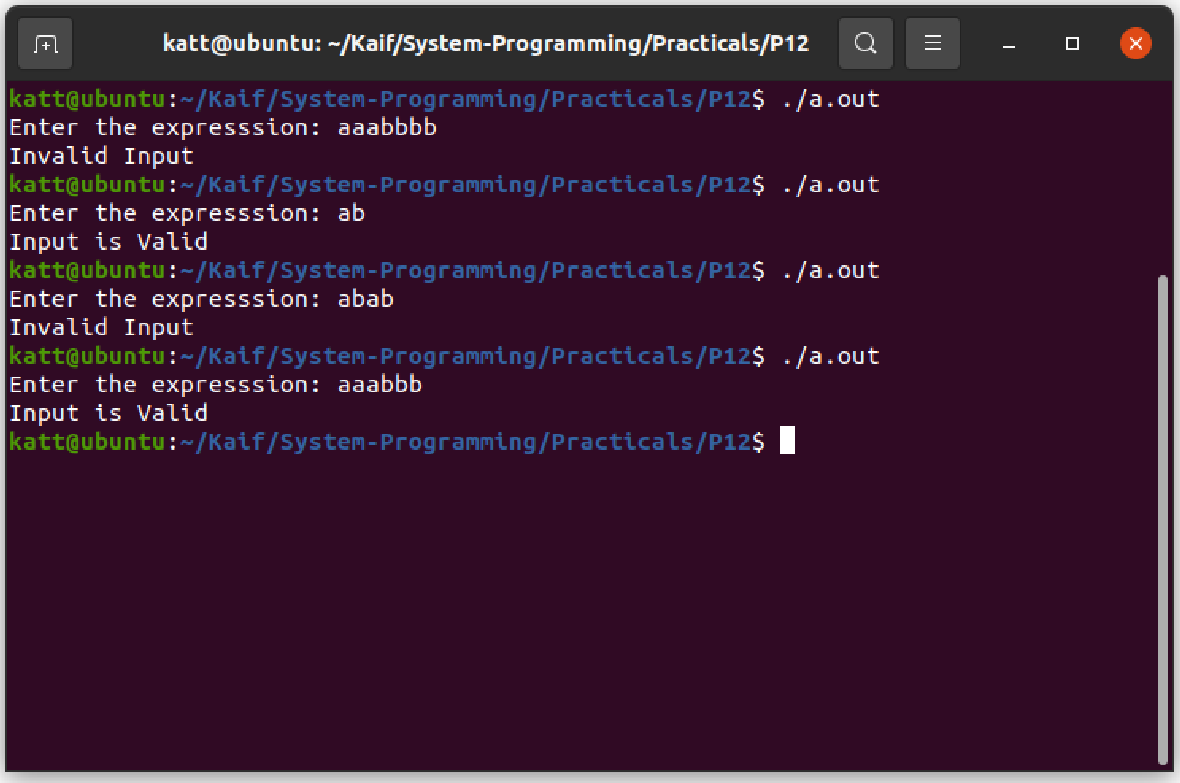
Lex file

```
%{
#include "y.tab.h"
%}
alpha [Aa]
beta [Bb]
newline [\n]
%%
{alpha} { return alpha ;}
{beta} {return beta;}
{newline} { return newline ;}
. { printf("Invalid Expression\n");exit(0); }
%%
```

Yacc file

```
%{
#include<stdio.h>
#include<stdlib.h>
#include<strings.h>
%}
%token alpha beta newline
%%
line : term newline {printf("Input is Valid\n"); exit(0);};
term: alpha term beta | ;
%%
```

```
int yyerror(char *msg)
{
printf("Invalid Input\n");
exit(0);
}

int main ()
{
printf("Enter the expresssion: ");
yyparse();
}
```



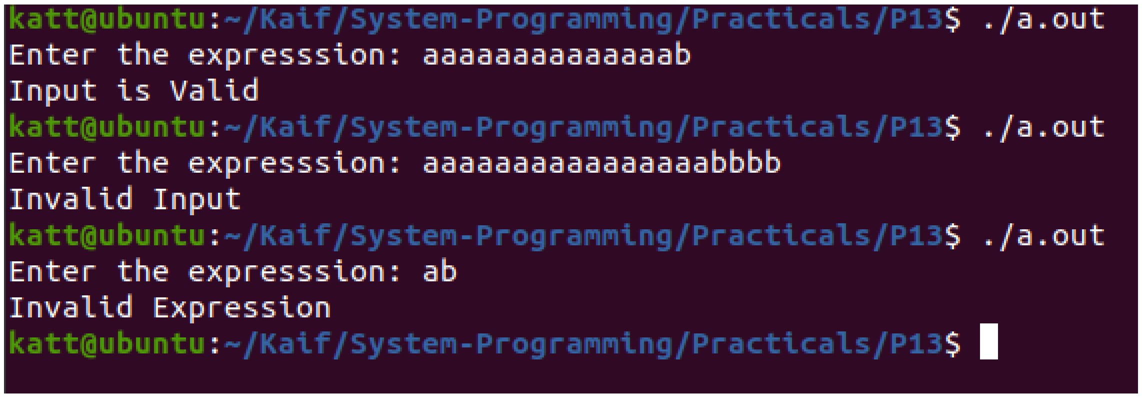
```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P12$ ./a.out
Enter the expresssion: aaabbbb
Invalid Input
katt@ubuntu:~/Kaif/System-Programming/Practicals/P12$ ./a.out
Enter the expresssion: ab
Input is Valid
katt@ubuntu:~/Kaif/System-Programming/Practicals/P12$ ./a.out
Enter the expresssion: abab
Invalid Input
katt@ubuntu:~/Kaif/System-Programming/Practicals/P12$ ./a.out
Enter the expresssion: aaabbb
Input is Valid
katt@ubuntu:~/Kaif/System-Programming/Practicals/P12$
```

13. Program in YACC to recognize the language (anb , n>=10). (Output to say input is valid or not)

Lex File
```
%{
#include "y.tab.h"
%}
alpha [a]{10,}
beta [b]
newline [\n]
%%
{alpha} { return alpha ;}
{beta} {return beta;}
{newline} { return newline ;}
. { printf("Invalid Expression\n");exit(0); }
%%
```

Yacc file
```
%{
#include<stdio.h>
#include<stdlib.h>
#include<strings.h>
%}
%token alpha beta newline
%%
line : term beta newline {printf("Input is Valid\n"); exit(0);};
term: alpha term |;
```

```
%%

int yyerror(char *msg)
{
printf("Invalid Input\n");
exit(0);
}

int main ()
{
printf("Enter the expresssion: ");
yyparse();
}
```



```
katt@ubuntu:~/Kaif/System-Programming/Practicals/P13$ ./a.out
Enter the expresssion: aaaaaaaaaaaaaab
Input is Valid
katt@ubuntu:~/Kaif/System-Programming/Practicals/P13$ ./a.out
Enter the expresssion: aaaaaaaaaaaaaaaabbbb
Invalid Input
katt@ubuntu:~/Kaif/System-Programming/Practicals/P13$ ./a.out
Enter the expresssion: ab
Invalid Expression
katt@ubuntu:~/Kaif/System-Programming/Practicals/P13$
```