PART-A

Execute of the following programs using LEX:

1. *Program to count the number of vowels and consonants in a given string.*

```
/*PROGRAM TO COUNT NO OF VOWELS AND CONSONANTS IN A GIVEN STRING*/

%{
  int vo=0;
  int co=0;
%}

%%

[ \t\n]+
[aeiouAEIOU] vo++;
[a-zA-Z] co++;

%%

main()
   {
    printf("\nEnter any string : ");
    yylex();
    printf("The No of Vowels     = %d\n",vo);
    printf("The No of Consonants = %d\n",co);
   }
```

OUTPUT:

```
==================================================================

[Anil@localhost Anil]$ vi lab1.l
[Anil@localhost Anil]$ lex lab1.l
[Anil@localhost Anil]$ cc lex.yy.c -ll
[Anil@localhost Anil]$ ./a.out

Enter any string : I Like You Anil   (press crtl-d)
The No of Vowels     = 7
The No of Consonants = 7

==================================================================
```

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

```
/*PROGRAM TO COUNT NO OF CHARACTERS,WORDS,SPACES & LINES IN A
GIVEN INPUT FILE*/


%{
  int cc=0;
  int wc=0;
  int lc=0;
  int bc=0;
%}

word [^ \t\n]+
eol [\n]
blank [ ]

%%

{blank} bc++;
{word} {wc++;cc+=yyleng;}
{eol} {cc++;lc++;}
. {ECHO;cc++;}

%%

main(int argc,char* argv[])
   {
    yyin=fopen(argv[1],"r");
    yylex();
    printf(" Character count = %d\n Word count       = %d\n Line
count      = %d\n Blank count     = %d\n",cc,wc,lc,bc);
   }
```

OUTPUT:

```
=====================================================================

[Anil@localhost Anil]$ vi lab2.l
[Anil@localhost Anil]$ lex lab2.l
[Anil@localhost Anil]$ cc lex.yy.c -ll
[Anil@localhost Anil]$ ./a.out inputF2
 Character count = 31
 Word count      = 8
 Line count      = 2
 Blank count     = 6

[Anil@localhost Anil]$ cat inputF2
Hi dear, How are you?
Miss you dear.

=====================================================================
```

3. *Program to count number of - a)Positive and Negative integers b)Positive and Negative fractions*

```
/*PROGRAM TO NO OF POSITIVE & NEGATIVE INTEGERS AND AS WELL
FRACTIONS*/

%{
  int pc=0;
  int nc=0;
  int pf=0;
  int nf=0;
%}

%%

[0-9]+ pc++;
-?[0-9]+ nc++;
([0-9]+\.[0-9]+) pf++;
-?([0-9]+\.[0-9]+) nf++;
. ECHO;

%%
main()
   {
    printf("\nGive the input (integers/float) : ");
    yylex();
    printf("No of +ve integer numbers     = %d\n",pc);
    printf("No of -ve integer numbers     = %d\n",nc);
    printf("No of +ve floating pt numbers = %d\n",pf);
    printf("No of -ve floating pt numbers = %d\n",nf);
   }
```

OUTPUT:

```
==========================================================================
[Anil@localhost Anil]$ vi lab3.l
[Anil@localhost Anil]$ lex lab3.l
[Anil@localhost Anil]$ cc lex.yy.c -ll
[Anil@localhost Anil]$ ./a.out

Give the input (integers/float) : -4 -0.2 0.4 5  (press crtl-d)

No of +ve integer numbers     = 1
No of -ve integer numbers     = 1
No of +ve floating pt numbers = 1
No of -ve floating pt numbers = 1

==========================================================================
```

4. *Program to count the numbers of comments lines in a given c program. Also eliminate them and copy that program into separate file.*

```
/*PROGRAM TO COUNT NO OF COMMENT LINES IN A GIVEN C PRGM & COPY
THAT PRGM TO SEPARATE FILE*/


%{
  int cl=0;
%}

%%

[ \t]*"//"[ \t]*.* {fprintf(yyout," ");cl++;}
[  \t]*"/*"([  ]*.*[  \t]*)*[\n]*.*[  ]*"*/"\n  {fprintf(yyout,"
");cl++;}
[ \t]*.*[\t]* {fprintf(yyout,"%s",yytext);}

%%

main(int argc, char* argv[])
   {
    FILE *file,*file1;
    if(argc>2)
      {
       file = fopen(argv[1],"r");
       file1 = fopen(argv[2],"w");
       yyin = file;
       yyout = file1;
       yylex();
       printf("NO OF COMMENTS = %d\n",cl);
      }
    else
        {
         printf("INSUFFICIENT ARGUMENTS,PASS THE ARGUMENTS\n");
        }
   }
```

OUTPUT:

```
[Anil@localhost Anil]$ vi lab4.l
[Anil@localhost Anil]$ lex lab4.l
[Anil@localhost Anil]$ cc lex.yy.c -ll
[Anil@localhost Anil]$ ./a.out
 INSUFFICIENT ARGUMENTS,PASS THE ARGUMENTS

[Anil@localhost Anil]$ ./a.out inputF4 outputF4
NO OF COMMENTS = 4

[Anil@localhost Anil]$ cat inputF4
/*just an c example*/
#include<stdio.h>
#include<conio.h>
void main()
   {
    int a,b;
    printf("Enter any two integer values : ");
    scanf("%d",&a,&b);
    //its not exactly need
    a = a + b;
    b = b;
    /*testing*/
    printf("Value of a & b = %d & %d \n",a,b);
    //end
    getch();
   }

[Anil@localhost Anil]$ cat outputF4
 #include<stdio.h>
 #include<conio.h>
void main()
   {
    int a,b;
    printf("Enter any two integer values : ");
    scanf("%d",&a,&b);

    a = a + b;
    b = b;
     printf("Value of a & b = %d & %d \n",a,b);

    getch();
   }
```

5. *Program to count the number of 'scanf' and 'printf' statements in a C program. Replace them with 'readf' and 'writef' statements respectively.*

```
/*PROGRAM TO COUNT NO OF printf'S & scanf'S & REPLACE THEM WITH
readf & writef respectively*/


%{
  int pc=0,sc=0;
%}

%%

printf fprintf(yyout,"WRITE");pc++;
scanf fprintf(yyout,"READ");sc++;
. ECHO;

%%

main(int argc,char* argv[])
   {
    if(argc!=3)
      {
       printf("INSUFFICENT,pass the file names\n");
      }
     yyin=fopen(argv[1],"r");
     yyout=fopen(argv[2],"w");
     yylex();
     printf("   NUMBER   OF   printf'S   =   %d\n   NUMBER   OF
scanf'S=%d\n",pc,sc);
   }
```

OUTPUT:

```
==========================================================================

[Anil@localhost Anil]$ vi lab5.l
[Anil@localhost Anil]$ lex lab5.l
[Anil@localhost Anil]$ cc lex.yy.c -ll
[Anil@localhost Anil]$ ./a.out inputF4 outputF5
 NUMBER OF printf'S = 2
 NUMBER OF scanf'S=1

[Anil@localhost Anil]$ cat inputF4
/*just an c example*/
#include<stdio.h>
#include<conio.h>
void main()
   {
    int a,b;
    printf("Enter any two integer values : ");
    scanf("%d",&a,&b);
    //its not exactly need
    a = a + b;
    b = b;
    /*testing*/
    printf("Value of a & b = %d & %d \n",a,b);
    //end
    getch();
   }

[Anil@localhost Anil]$ cat outputF5
/*just an c example*/
#include<stdio.h>
#include<conio.h>
void main()
   {
    int a,b;
    WRITE("Enter any two integer values : ");
    READ("%d",&a,&b);
    //its not exactly need
    a = a + b;
    b = b;
    /*testing*/
    WRITE("Value of a & b = %d & %d \n",a,b);
    //end
    getch();
   }
==========================================================================
```

6. *Program to recognize a valid arithmetic expression and identify the identifiers and operators present. Print them separately.*

```
/*TO RECOGNIZE A VALID ARITHEMATIC EXPRSN & IDENTIFY THE
IDENFIERS & OPERAT0RS PRESENT,PRINT THEM SEPARATELY*/

%{
  #include<stdio.h>
  int oc=0,valid=0,i=0;
%}

num[0-9]+
ope[*/+-]

%%
{num}({ope}{num})+ {
valid=1;
for(i=0;yytext[i];i++)
   {
   switch(yytext[i])
        {
         case '+':
         case '-':
         case '*':
         case '/':oc++;
        }
   }
}

{num}\n {printf("\nONLY A NUMBER GIVEN,give an expression\n");
        // exit(0);
        }
\n { if(valid)
       {
        printf("\nVALID OPERATOR COUNT = %d\n",oc);
        exit(0);
       }
      else
          printf("\nINVALID INPUT \nENTER THE EXPRESSION \n");
          valid=oc=0; i=0;}
%%
main()
   {
    printf("ENTER THE EXPRESSION\n");
    yylex();
   }
```

OUTPUT:

```
==========================================================================

[Anil@localhost Anil]$ vi lab6.l
[Anil@localhost Anil]$ lex lab6.l
[Anil@localhost Anil]$ cc lex.yy.c -ll
[Anil@localhost Anil]$ ./a.out
ENTER THE EXPRESSION
3

ONLY A NUMBER GIVEN,give an expression
q+o
q+o
INVALID INPUT
ENTER THE EXPRESSION
3+9-0

VALID OPERATOR COUNT = 2

==========================================================================
```

7. *Program to recognize and count the number of identifiers in a given input file.*

```
/*PROGRAM TO RECONGNIZE & COUNT THE NO OF IDENTIFIERS IN A GIVEN
INPUT FILE*/

%{
  int count=0;
%}

%%

("int")|("float")|("double")|("char") {
int ch;
ch=input();
for(;;)
  {
   if(ch==',')
     count++;
   else if(ch==';')
        break;
        ch=input();
  }
  count++;
}

%%

main(int argc,char* argv[])
   {
    if(argc>1)
      {
       yyin=fopen(argv[1],"r");
       yylex();
       printf("NO OF IDENTIFIERS = %d\n",count);
      }
    else printf("\nPass valid file name as an ARGUMENT\n");
  }
```

OUTPUT:

```
======================================================================

[Anil@localhost Anil]$ vi lab7.l
[Anil@localhost Anil]$ lex lab7.l
[Anil@localhost Anil]$ cc lex.yy.c -ll
[Anil@localhost Anil]$ ./a.out inputF7

NO OF IDENTIFIERS = 8

[Anil@localhost Anil]$ ./a.out

Pass valid file name as an ARGUMENT

[Anil@localhost Anil]$ cat inputF7
int a,b,c;
float x,y,z;
char a[],b;

======================================================================
```

PART-B

Execute of the following programs using YACC:

*1) Program to test the validity of a simple expression involving operators +, -, * and /.*

```
/* Lex Program that passes token */

%{
#include"y.tab.h"
%}
%%

[a-z0-9]+ {return num;}
[+/] {return op;}
[*-] {return op1;}
%%

/* Yacc program to check for valid expression */

%{
#include<stdio.h>
extern int yyerror();
extern int yylex();
%}
%token num op op1
%%

st:exp {printf("valid");exit(0);}
exp:exp op num
|exp op1 num
|num op num
|num op1 num
;
%%

main()
 {
   yyparse();
 }
yyerror()
 {
  printf("Invalid");
 }
```

OUTPUT

```
==========================================================

[Anil@Anil ~]$ lex 1.l
[Anil@Anil ~]$ yacc -d 1.y
[Anil@Anil ~]$ cc lex.yy.c y.tab.c -ll
[Anil@Anil ~]$ ./a.out
a+b

valid

[Anil@Anil ~]$ ./a.out
a

Invalid

==========================================================
```

2) Program to recognize nested IF control statements and display the number of levels of nesting.

```
/* Lex program that passes token */

%{
#include"y.tab.h"
%}
num[0-9]
alp[a-z]
id {alp}
({num}|{alp})
bin[*/+-]
not[!]
eq[=]
%%

"if" {return ff;}
{num} {return num;}
{alp} {return alp;}
{id} {return id;}
{bin} {return bin;}
{not} {return not;}
{eq} {return eq;}
("++")|("--") {return inc;}
("==")|("<")|("<=")|(">")|(">=") {return rel;}
. {return yytext[0];}
%%


/* Yacc program to recognise if control statement */

%{
 #include<stdio.h>
 int count=0;
%}
%token ff alp id bin not num eq rel inc
%%
st: com_nest {printf("\nValid no of nesting : %d\n",count);
exit(0);}
;
com_nest: nest {count++;}
;
nest:ff '('condi')' one_st
|ff '('condi')' '{'many_st'}'
|ff '('condi')' '{'nest'}' {count++;}
;
```

```
condi:condi rel id
|condi rel num
|num
|id
|alp
;
one_st:id eq id bin id';'
|alp bin alp';'
|alp inc';'
|id inc';'
;
many_st:many_st one_st
|one_st one_st
;
%%

main()
 {
  yyparse();
 }
yyerror()
 {
  printf("\nInvalid\n");
 }
```

OUTPUT

```
============================================================

[Anil@Anil ~]$ lex ex2.l
[Anil@Anil ~]$ yacc -d ex2.y
[Anil@Anil ~]$ cc lex.yy.c y.tab.c -ll
[Anil@Anil ~]$ ./a.out
if(a>3){if(b==3)c++;}

Valid no of nesting : 2

[Anil@Anil ~]$ ./a.out

if(a=1)a--;

Invalid

============================================================
```

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

```
/* Lex Program to send tokens to the yacc program */

%{
#include"y.tab.h"
%}
%%
[a-z] {return alp;}
[0-9] {return dig;}
. {return yytext[0];}

%%

/* Yacc Program to validate the given variable */

%{
#include<stdio.h>
extern int yyparse();
extern int yylex();
%}
%token dig alp
%%
st:id {printf("Valid\n"); exit(0);}
;
id:id dig
|id alp
|alp
;
%%

main()
 {
  yyparse();
 }
yyerror()
{
 printf("Invalid\n");
}
```

OUTPUT

```
==========================================================

[Anil@Anil ~]$ lex 3.l
[Anil@Anil ~]$ yacc -d 3.y
[Anil@Anil ~]$ cc lex.yy.c y.tab.c -ll
[Anil@Anil ~]$ ./a.out

a120

Valid

[Anil@Anil ~]$ ./a.out

20

Invalid

==========================================================
```

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

```
/* Lex  program that passes tokens */

 %{
 #include<"y.tab.h"
 extern int yyval;
 %}

 %%

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

 %%

/* Yacc program to evaluate the expression */

 %{
 #include<stdio.h>
 #include<ctype.h>
 %}
 %token dig
 %left '+' '-'
 %left '*' '/'

 %%

 st:exp{printf("valid:%d",$1);exit(0);}
 ;
 exp:exp '+' exp  {$$=$1+$3;}
 exp '-' exp   {$$=$1-$3;}
 |exp '*' exp   {$$=$1*$3;}
 |exp '/' exp   {if($3==0) printf("error"); else  $$=$1/$3;}
 |dig {$$=$1;}
 |'('exp')' {$$=$2;}
 ;

 %%

 main()
  {
   yyparse();
  }
```

```
  yyerror()
   {
    printf("Invalid");
   }
```

OUTPUT

==========

```
[Anil@Anil ~]$ yacc -d 4.y
[Anil@Anil ~]$ cc y.tab.c -ll
[Anil@Anil ~]$ ./a.out
```

10+10-10*10

Valid: -80

==========

5) Program to recognize the grammar (an b, n>=10).

```
/*lex program*/

%{
 #include"y.tab.h"
%}
%%

[a] {return a1;}
[b] {return b1;}
%%

/*yacc program*/
%{
 #include<stdio.h>
int count=10;
%}
%token a1 b1
%%

st:s{printf("valid\n");}
;
s:a1 a1 a1 a1 a1 a1 a1 a1 a1 a1 rab b1
;
rab:rab a1
|
;
%%

main()
{
 yyparse();
}

yyerror()
 {
  printf("Invalid");
 }
```

OUTPUT

```
============================================================

[Anil@Anil ~]$ lex yc7.l
[Anil@Anil ~]$ yacc -d yc7.y
[Anil@Anil ~]$ cc lex.yy.c y.tab.c -ll
[Anil@Anil ~]$ ./a.out

aaaaaaaaaaab

valid

aaaab

Invalid

============================================================
```