# SYSTEM SOFTWARE PROGRAMS

Execution of the following programs using LEX:

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

```
%{
#include<stdio.h>
int vc=0,cc=0;
%}

%%
[aeiouAEIOU] vc++;
[a-zA-Z] cc++;
[ \n\t] ;
%%

int yywrap()
{
        return 1;
}

main()
{
        printf("enter a string\n");
        yylex();
        printf("no. of vowels=%d\n no of consonant=%d\n",vc,cc);
}
```

```
***********output***********

$ lex pgm1.l
$ gcc -o pgm1 lex.yy.c
$ ./pgm1
enter a string
uvce is our college
no. of vowels=8
no of consonant=8
```

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

```
%{
#include<stdio.h>
int cc=0,wc=0,sc=0,lc=0;
%}

%%
[^ \n\t]+ wc++,cc+=yyleng;
[ ] sc++,cc++;
[\n] lc++;
[\t] sc+=8,cc+=8;
%%


int yywrap()
{
  return 1;
}

main()
{
        char fname[10];
        printf("enter the file name\n");
        scanf("%s",fname);
        yyin=fopen(fname,"r");
        yylex();
        printf("character=%d\n wrds=%d\n spaces=%d\n
lines=%d\n",cc,wc,sc,lc);
}
```

**********output**********

```
$ lex pgm2.l
$ gcc -o pgm2 lex.yy.c
$ ./pgm2
enter the file name
ex.txt
character=154
wrds=29
spaces=23
lines=9
```

3. Program to count the (i) positive and negative integers (ii) positive and negative fractions.

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

int yywrap()
{
return 1;
}

void main()
{
        printf("enter the number");
        yylex();
        printf("+ve i=%d\n-ve i=%d\n+ve f=%d\n-ve f=%d",pi,ni,pf,nf);
}
```

**********output*********

```
$ lex pgm3.l
$ gcc -o pgm3 lex.yy.c
$ ./pgm3
enter the number-1

2.5

2.3

2

-2.33

+ve i=1
-ve i=1
+ve f=2
-ve f=1
```

4. Program to count the number of comment lines in a given C program.Also eliminate them and copy that program into separate file.

```
%{
#include<stdio.h>
int count;
%}

%%
"/*"[a-zA-Z0-9' '\n\t]*"*/" count++;
"//"[^ \n]* count++;
%%

int yywrap()
{
return 1;
}

main()
{
        char fname1[10],fname2[10];
        printf("enter the file1");
        scanf("%s",fname1);
        yyin=fopen(fname1,"r");
        printf("enter the file2");
        scanf("%s",fname2);
        yyout=fopen(fname2,"w");
        yylex();
        printf("no of cmnts=%d\n",count);
}
```

**********output**********

```
$ lex pgm4.l
$ gcc -o lex.yy.c
$ ./pgm4
enter the file1 ex.txt
enter the file2 ex2.txt
no of cmnts=3
```

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

```
%{
#include<stdio.h>
int pf=0,sf=0;
%}

%%
"printf" {fprintf(yyout,"writef"); pf++;}
"scanf" {fprintf(yyout,"readf"); sf++;}
%%

yywrap()
{
return 1;
}

main()
{
char fname1[10],fname2[10];
printf("enter file1");
scanf("%s",fname1);
yyin=fopen(fname1,"r");
printf("enter file2");
scanf("%s",fname2);
yyout=fopen(fname2,"w");
yylex();
printf("printf=%d\nscanf=%d\n",pf,sf);
}
```

**********output*********

```
$ lex pgm5.l
$ gcc -o pgm5 lex.yy.c
$ ./pgm5
enter file11.c
enter file2ex2.txt
printf=2
scanf=0
```

Execution of the following programs using YACC:

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

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

%%
[a-zA-Z][a-zA-Z0-9]* return ID;
[0-9]+ return NUM;
[\n] return NL;
. return yytext[0];
%%

int yywrap()
{
return 1;
}

%{
#include<stdio.h>
#include<stdlib.h>
%}
%token NUM ID NL
%left '*''/'
%left '+''-'

%%
stmt:exp NL {printf("valid\n");exit(0);}
exp : exp'+'exp
     | exp'-'exp
     | exp'*'exp
     | exp'/'exp
     | '('exp')'
     | '['exp']'
     | '{'exp'}'
     | NUM
     | ID
     ;
%%

main()
{
        printf("enter expression\n");
        yyparse();
}
yyerror()
{
        printf("invalid\n");
        exit(0);
}
```

```
***********output***********
$ yacc -d 1.y
$ cc y.tab.c lex.yy.c
$ ./a.out
enter expression
a+[b-(c+)*d]
invalid
$ ./a.out
enter expression
a+[b-(c+d)*e]
valid
```

2. Program to recognize a valid arithmetic expression that user operators +,-,*,/.

```
%{
#include "y.tab.h"
%}
%%
[0-9]|[0-9]*"."[0-9]+ return NUM;
[\n] return NL;
. return yytext[0];
%%
int yywrap()
{
        return 1;
}
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token NUM NL
%left '*''/'
%left '+''-'
%%
stmt:exp NL {printf("valid\n");exit(0);}
     ;exp: exp'+'exp
    | exp'-'exp
    | exp'*'exp
    | exp'/'exp
    | '('exp')'
    | '{'exp'}'
    | '['exp']'
    | NUM
    ;%%
main()
{
        printf("enter the exp\n");
        yyparse();
}
yyerror()
{
        printf("invalid\n");
        exit(0);
}
```

**********output**********

```
$ yacc -d 3.y
$ cc y.tab.c lex.yy.c
$ ./a.out
enter the exp
1+7-5*(4+4)/7
valid
$ ./a.out
enter the exp 1++
invalid
```

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

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

%%

"if" return IF;
[0-9]+ return NUM;
[a-zA-Z][a-zA-Z0-9]* return ID;
[*/+-] return BIN;
[=] return EQU;
"++"|"--" return INC;
"=="|"<"|">"|"<="|">="|"!=" return REL;
. return yytext[0];

%%

int yywrap()
{
        return 1;
}

$ gedit 2.y

%{
#include<stdio.h>
#include<stdlib.h>
int count=0;
%}

%token IF NUM ID BIN EQU INC REL

%%
st: com_nest {printf("valid\nno. of nesting:%d\n",count);exit(0);}
   ;

com_nest: nest {count++;}
        ;

nest: IF'('cond')' one_st
      IF'('cond')' '{'many_st'}'
      IF'('cond')' '{'nest'}' {count++;}
      ;
cond: ID REL ID
     | ID REL NUM
     | NUM REL NUM
     | ID
     | NUM
     ;
```

```
one_st: ID EQU ID BIN ID';'
      | ID EQU ID BIN NUM';'
      | ID EQU NUM BIN NUM';'
      | ID EQU NUM';'
      | ID INC';'
      ;

many_st: many_st one_st
       | one_st one_st
       ;

%%

main()
{
   printf("enter the statement\n");
   yyparse();
}

yyerror()
{
   printf("invalid\n");
   exit(0);
}
```

*********output**********

```
$ lex 2.l
$ yacc -d 2.y
$ cc y.tab.c lex.yy.c
$ ./a.out
enter the statement if(a=b){if(1){exit()}}
no. of nesting:2
enter the statement if(a=b){exit()}
no. of nesting:1
enter the statement k
no. of nesting:0
```