Compiler Construction Lab

Q1. Write a program using LEX to recognize a valid arithmetic expression and to recognize the identifiers and operators present. Print them separately.

Source Code->

```
%{
#include<stdio.h>
int a=0,s=0,m=0,d=0,ob=0,cb=0;
int flaga=0, flags=0, flagm=0, flagd=0;
%}
id [a-zA-Z]+
%%
{id} {printf("\n %s is an identifier\n",yytext);}
[+] {a++;flaga=1;}
[-] {s++;flags=1;}
[*] {m++;flagm=1;}
[/] {d++;flagd=1;}
[(] {ob++;}
[)] {cb++;}
%%
int main()
{
printf("Enter the expression\n");
yylex();
if(ob-cb==0)
{
printf("Valid expression\n");
}
else
{
printf("Invalid expression");
}
printf("\nAdd=%d\nSub=%d\nMul=%d\nDiv=%d\n",a,s,m,d);
printf("Operators are: \n");
if(flaga)
printf("+\n");
if(flags)
printf("-\n");
if(flagm)
printf("*\n");
if(flagd)
printf("/\n");
return 0;
}
```

Output:->

```
$lex p2a.l
$cc lex.yy.c –ll
$./a.out
```

```
Enter the expression
(a+b*c)
a is an identifier
b is an identifier
c is an identifier
[Ctrl-d]
Valid expression
Add=1
Sub=0
Mul=1
Div=0
Operators are:
+
*
```

Q2.Write a program using LEX to recognize whether a given sentence is simple or compound.

Source Code->

```
%{
int flag=0;
%}
%%
(""[aA][nN][dD]"")|(""[oO][rR]"")|(""[bB][uU][tT]"") {flag=1;}
%%
int main()
{
printf("Enter the sentence\n");
yylex();
if(flag==1)
printf("\nCompound sentence\n");
else
printf("\nSimple sentence\n");
return 0;
}
```

OUTPUT:->

```
$lex p2b.l
$cc lex.yy.c –ll
$./a.out
Enter the sentence
I am Pooja
I am Pooja
[Ctrl-d]
Simple sentence
$./a.out
Enter the sentence
CSE or ISE
CSE or ISE
```

[Ctrl-d]
Compound sentence

Q3. Write a program using LEX to recognize and count the number of identifiers in a given input file.

Source Code:->

```
%{
#include<stdio.h>
int count=0;
%}
op [+-*/]
letter [a-zA-Z]
digitt [0-9]
id {letter}*|({letter}{digitt})+
notid ({digitt}{letter})+
%%
[\t\n]+
("int")|("float")|("char")|("case")|("default")| ("if")|("for")|("printf")|("scanf") {printf("%s is a keyword\n", yytext);}
{id} {printf("%s is an identifier\n", yytext); count++;}
{notid} {printf("%s is not an identifier\n", yytext);}
%%
int main()
{
FILE *fp;
char file[10];
printf("\nEnter the filename: ");
scanf("%s", file);
fp=fopen(file,"r");
yyin=fp;
yylex();
printf("Total identifiers are: %d\n", count);
return 0;
}
```

Output:->

```
$cat > input
int
float
78f
90gh
a
d
are case
default
printf
scanf
$lex p3.l
$cc lex.yy.c –ll
$./a.out
Enter the filename: input
int is a keyword
```

```
float is a keyword
78f is not an identifier
90g is not an identifier
h is an identifier
a is an identifier
d is an identifier
are is an identifier
case is a keyword
default is a keyword
printf is a keyword
scanf is a keyword
total identifiers are: 4
```

Q4.Write a LEX program to count the numbers of comment lines in a given C program. Also eliminate them and copy the resulting program into separate file.

Source code:->

```
%{
int com=0;
%}
%%
"/*"[^\n]+"*/" {com++;fprintf(yyout, " ");}
%%
int main()
{
printf("Write a C program\n");
yyout=fopen("output", "w");
yylex();
printf("Comment=%d\n",com);
return 0;
}
```

Ouput:->

```
$lex p1b.l
$cc lex.yy.c –ll
$./a.out
      Write a C program
                  #include<stdio.h>
                    int main()
                     {
                      int a, b;
                        /*float c;*/
                       printf(“Hai”);
                       /*printf(“Hello”);*/
                         }
[Ctrl-d]
Comment=2
$cat output
#include<stdio.h>
```

```
int main()
{
   int a, b;
   printf(“Hai”);
}
```

Q5.Write a program using LEX to count the number of characters, words, spaces and lines in a given input file.

Source Code-> {provided file should present in that directory or create it}

```
%{
int ch=0, bl=0, ln=0, wr=0;
%}
%%
[\n] {ln++;wr++;}
[\t] {bl++;wr++;}
[" "] {bl++;wr++;}
[^\n\t] {ch++;}
%%
int main()
{
FILE *fp;
char file[10];
printf("Enter the filename: ");
scanf("%s", file);
yyin=fp;
yylex();
printf("Character=%d\nBlank=%d\nLines=%d\nWords=%d", ch, bl, ln, wr);
return 0;
}
```

Output ->

```
$cat > input
Girish rao salanke

$lex p1a.l
$cc lex.yy.c –ll
$./a.out
Enter the filename: input
Character=16
Blank=2
Lines=1
Word=3
```

Q6. Program to find whether given number is Octal or Hexadecimal.

```
%{
        /*first program*/
%}
Oct [o][0-9]+
```

```
Hex [o][x|X][0-9A-F]+
%%
{Hex} printf("this is hexadecimal number");
{Oct} printf("this is an octal number");
%%
main()
{
yylex();
}
int yywrap()
{
return 1;
}

$ ./a.out
o6567
this is an octal number
oX2780ad
this is hexadecimal numberad
oX58976AAAD
this is hexadecimal number
```

Q7.Program a C program to compute the FIRST of a given grammar.

```
#include<stdio.h>
#include<ctype.h>
int main()
{
	int i,n,j,k;
	char str[10][10],f;
	printf("enter the number of productions\n");
	scanf("%d",&n);
	printf("enter grammar\n");
	for(i=0;i<n;i++)
		scanf("%s",&str[i]);
	for(i=0;i<n;i++)
	{
		f=str[i][0];
		int temp=i;
		if(isupper(str[i][3]))
		{
repeat:
		for(k=0;k<n;k++)
		{
			if(str[k][0]==str[i][3])
			{
				if(isupper(str[k][3]))
				{
					i=k;
```

```
                        goto repeat;
                    }
                    else
                    {
                        printf("First(%c)=%c\n",f,str[k][3]);
                    }
                }
            }
            }
            else
            {
                printf("First(%c)=%c\n",f,str[i][3]);
            }
            i=temp;
        }
}
```

OUTPUT:

```
$ ./a.out
enter the number of productions
3
enter grammar
S->AB
A->a
B->b
First(S)=a
First(A)=a
First(B)=b
```

Q8. Lex program to recognize keywords and identifiers.

```
%{
/*LEX to recognize keywords and identifiers */
%}

letter [a-zA-Z]+[a-zA-Z0-9]*
notvalid [0-9]+[a-zA-Z]+[a-zA-Z0-9]*
digit [0-9]*
%%
int|float|char|double|else|for|if|while|main|printf {printf("\nreserved words is %s",yytext);}
{letter} {printf("valid identifiers is %s \n",yytext);}
{notvalid} {printf("invalid identifier is %s \n",yytext);}
{digit} {printf("number is %s \n",yytext);}
%%
main(int argc,char **argv)
{
if(argc>1)
yyin=fopen(argv[1],"r");
else
yyin=stdin;
yylex();
printf("\n");
```

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

Q9. Lex program to count number of vowels and consonant

```
%{
 int v=0,c=0;
%}
%%
[aeiouAEIOU] v++;
[a-zA-Z] c++;
%%
main()
{
printf("ENTER INTPUT : \n");
yylex();
printf("VOWELS=%d\nCONSONANTS=%d\n",v,c);
}
```

10. Lex program to count the type of numbers

```
%{
int pi=0,ni=0,pf=0,nf=0;
%}
%%
\+?[0-9]+ pi++;
\+?[0-9]*\.[0-9]+ pf++;
\-[0-9]+ ni++;
\-[0-9]*\.[0-9]+ nf++;
%%
main()
{
printf("ENTER INPUT : ");
yylex();
printf("\nPOSITIVE INTEGER : %d",pi);
printf("\nNEGATIVE INTEGER : %d",ni);
printf("\nPOSITIVE FRACTION : %d",pf);
printf("\nNEGATIVE FRACTION : %d\n",nf);
}
```

11. Lex program to count the number of printf and scanf statements

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

```
pf++;
fprintf(yyout,"%s","writef");
}
scanf {
sf++;
fprintf(yyout,"%s","readf");
}
%%
main()
{
yyin=fopen("file1.l","r+");
yyout=fopen("file2.l","w+");
yylex();
printf("NUMBER OF PRINTF IS %d\n",pf);
printf("NUMBER OF SCANF IS %d\n",sf);
}
```

12. Lex program to find simple and compound statements

```
%{
}%
%%
"and"|
"or"|
"but"|
"because"|
"nevertheless" {printf("COMPOUNT SENTANCE"); exit(0); }
. ;
\n return 0;
%%
main()
{
prntf("\nENTER THE SENTANCE : ");
yylex();
printf("SIMPLE SENTANCE");
}
```

Q13. Lex program to count the number of identifiers

```
%{
#include<stdio.h>
int id=0,flag=0;
%}
%%
"int"|"char"|"float"|"double" { flag=1; printf("%s",yytext); }
";" { flag=0;printf("%s",yytext); }
[a-zA-Z][a-zA-z0-9]* { if(flag!=0) id++; printf("%s",yytext); }
[a-zA-Z0-9]*"="[0-9]+ { id++; printf("%s",yytext); }
[0] return(0);
%%
main()
{
```

```
 printf("\n *** output\n");
 yyin=fopen("f1.l","r");
 yylex();
 printf("\nNUMBER OF IDENTIFIERS = %d\n",id);
 fclose(yyin);
 }
 int yywrap()
{
return(1);
}
```

Q14.Lex program to count the number of words,characters,blank spaces and lines

```
%{
int c=0,w=0,l=0,s=0;
%}
%%
[\n] l++;
[' '\n\t] s++;
[^' '\t\n]+ w++; c+=yyleng;
%%
int main(int argc, char *argv[])
{
if(argc==2)
{
yyin=fopen(argv[1],"r");
yylex();
printf("\nNUMBER OF SPACES = %d",s);
printf("\nCHARACTER=%d",c);
printf("\nLINES=%d",l);
printf("\nWORD=%d\n",w);
}
else printf("ERROR");
}
```

Q15. Lex program to count the number of comment lines

```
%{
#include<stdio.h>
int cc=0;
%}
%%
"/*"[a-zA-Z0-9' '\t\n]*"*/" cc++;
"//"[a-zA-Z0-9' '\t]* cc++;
%%
main()
{
yyin=fopen("f1.l","r");
yyout=fopen("f2.l","w");
yylex();
fclose(yyin);
fclose(yyout);
```

```
printf("\nTHE NUMBER OF COMMENT LINES = %d\n",cc);
}
```

Q16. Lex program to check the validity of arithmetic statement

```
%{
#include<stdio.h>
int opr=0,opd=0;
int n;
%}
%%
[\+\-\*\/] { printf("OPERATORS ARE %s\n",yytext);
opr++;
}
[a-zA-Z]+ { printf("OPERANDS ARE %s\n",yytext);
opd++;
}
[0-9]+ { printf("OPERANDS ARE %s\n",yytext);
opd++;
}
[a-zA-Z]+\+\-\*\/[a-zA-Z]+ { n=0; }
[0-9]+\+\-\*\/[0-9]+ { n=0; }
%%
main()
{
printf("\nENTER THE EXPRESSION : \n");
yylex();
printf("\nNUMBER OF OPERATORS ARE %d",opr);
printf("\nNUMBER OF OPERANDS ARE %d",opd);
if((n==0)&&(opd==opr+1))
printf("\nVALID EXPRESSION\n");
else
printf("\nINVALID EXPRESSION\n");
}
```

Q17. Lex program to find the number of constants

```
%{
#include<stdio.h>
int cons=0;
%}
%%
[0-9]+ { printf("\n%s",yytext); cons++; }
. ;
%%
main(int argc,char *argv[])
{
if(argc==2)
{
yyin=fopen(argv[1],"r");
yylex();
```

```
printf("\nNUMBER OF CONSTANTS : %d\n",cons);
}
else
printf("\nERROR");
}
```

Q18. Program to count the numbers of comment lines in a given C program. Also eliminate them and copy the resulting program into separate file.

```
%{
int c=0,state=1;
%}

%%
“/*” { state=0;}
“*/” { c++; if (!state) state=1;}
{ if (state==1)
fprintf(yyout,”%s”,yytext);
}
%%

FILE * fp;
main(int argc,char ** argv)
{
if(argc<=1)
{
printf(“\nNo file”);
exit(1);
}

fp=fopen(argv[1],”w”);

if(!fp)
{
printf(“\nNo output file”);
exit(1);
}
yyout=fp;

fp=fopen(argv[1],”r”);

if(!fp)
{
printf(“\nNo inpput file”);
exit(1);
```

```
}
yyin=fp;
yylex();
printf("\nNumber of comment lines : %d",c);
}
yywrap()
{
if(state==0)
{
printf("\n Unterminated commennt");
return 1;
}
}
```

Q19. Program to recognize a valid arithmetic expression and to recognize the identifiers and operators present. Print them separately.

```
%{
#include<stdio.h>
#include<string.h>
#define max 20
int flag,i,j,k,top,b;
char stack[max],ident[max],oper[max],brac[max];
%}

%%
[a-zA-Z0-9] {j++;strcat(ident,yytext);}
[a-zA-Z0-9]+ {flag=1;}
"+" {oper[k++]='+';}
"-" {oper[k++]='-';}
"*" {oper[k++]='*';}
"/" {oper[k++]='*';}
"$" {oper[k++]='$';}
"^" {oper[k++]='^';}
"%" {oper[k++]='%';}
"(" { stack[++top]='(';brac[b++]='(';}
")" { if (stack[top]=='(' && top!=-1) top–;flag=0;brac[b++]=')';}
%%

int  main()
{
int i=j=k=b=flag=0;
top=-1;

printf("\nEnter the Expression : ");
yylex();
printf("\nThe identifiers are :  ");

for(i=0;i<j;i++)
```

```
printf("\t%c",ident[i]);

printf("\nNo.of identifiers are : %d",j);
printf("\nThe operators are :\n");

for(i=0;i<k;i++)
printf("\t%c",oper[i]);

printf("\nNo. of operators are :%d",k);
if(flag==0 && top==-1 && j==k+1)
printf("\nValid expression");
else
printf("\nInvalid expression");

return 0;
}
```

Q20. Program to recognize whether a given sentence is simple or compound.

```
%{
#include<stdio.h>
int F0=0,F1=0,F2=0,error=0,l1=0,l2=0;
%}

verb am|run|sit|did|study|is|large|go|come
subject [a-zA-Z]+
compnd "and"|"but"|"also"|"either"|"neither"|"yet"|"still"|"consequences"

%%
{verb} { if(F2==1)
l2=1;
F2=1;
if(F1==0)
error=1;
}

{compnd} { F0=1; }
{subject} { if(F1!=0)
l1=1;
F1++;
}
%%

main()
{
printf("\n Enter a sentence: ");
yylex();
```

```
if(error==1 || F2==0 || F1==0)
{
printf(“\n Invalid sentence”);
exit(0);
}

if(F0==1 && l1==1 && l2==1)
printf(“\n Compound sentence\n”);
else
printf(” \nSimple sentence\n”);
}
```

Q21. Write a lex program to count the number of comment lines in a given C program. Also eliminate them and copy that program into separate file

```
%{
#include
int comments=0;
%]

%%
"//".* {comments++;}
"/*"[a-zA-Z0-9\n]*"*/" {comments++;}
%%

main()
{
char s[10],d[10];
printf("Enter the source file and destination file\n");
scanf("%s%s",s,d);
yyin=fopen(s,"r"); /*open input file in read mode*/
yyout=fopen(d,"w"); /*open output file in write mode*/
yylex();
printf("Number of comments = %d\n",comments);
fclose(yyin);
fclose(yyout);
}
```

To compile save this file as cmt.lex
lex cmt.lex
cc yy.lex.c -ll
./a.out
Input the source file (any c program with comments)
Enter the destination file name as output.txt
To see the output:
cat output.txt