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
```


Scanned by CamScanner

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

```
%{
#include<stdio.h>
int vowels=0;
int cons=0;
%}
%%
[aeiouAEIOU] {vowels++;}
[a-zA-Z] {cons++;}
%%

int yywrap()
{
return 1;
}
int main()
{
printf("enter the string");
yylex();
printf("No of vowels=%d\n No of cons=%d", vowels,cons);
return 0;
}
```

Compile and Run the program:

```
[vizion@localhost ~]$ lex 1b.l
[vizion@localhost ~]$ cc lex.yy.c -ll
[vizion@localhost ~]$ ./a.out
enter the string Compiler Design

No of vowels=5
 No of cons=9[vizion@localhost ~]$
```

Scanned by CamScanner

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
```


Scanned by CamScanner

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
```

Scanned by CamScanner

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;
}
```


Scanned by CamScanner

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;
}
```

Scanned by CamScanner

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];}
D] {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:

Scanned by CamScanner

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
```

Scanned by CamScanner

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
```


Scanned by CamScanner

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
```

Scanned by CamScanner

COMPILER DESIGN PROGRAMS:

7. C Program to implement Pass 1 algorithm of assembler.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
FILE *f1,*f2,*f3,*f4;
int lc,sa,l,op1,o,len;
char m1[20],la[20],op[20],otp[20];
clrscr();
f1=fopen("input.txt","r");
f3=fopen("symtab.txt","w");
fscanf(f1,"%s %s %d",la,m1,&op1);
if(strcmp(m1,"START")==0)
{
sa=op1;
lc=sa;
printf("\t%s\t%s\t%d\n",la,m1,op1);
}
else
lc=0;
fscanf(f1,"%s %s",la,m1);
while(!feof(f1))
{
fscanf(f1,"%s",op);
printf("\n%d\t%s\t%s\t%s\n",lc,la,m1,op);
if(strcmp(la,"-")!=0)
{
fprintf(f3,"\n%d\t%s\n",lc,la);
}
f2=fopen("optab.txt","r");
fscanf(f2,"%s %d",otp,&o);
while(!feof(f2))
{
if(strcmp(m1,otp)==0)
{
lc=lc+3;
break;
}
fscanf(f2,"%s %d",otp,&o);
}
fclose(f2);
if(strcmp(m1,"WORD")==0)
{
lc=lc+3;
}
else if(strcmp(m1,"RESW")==0)
{
op1=atoi(op);
lc=lc+(3*op1);
```


Scanned by CamScanner

```
50.         }
51.         else if(strcmp(m1,"BYTE")==0)
52.         {
53.         if(op[0]=='X')
54.         lc=lc+1;
55.         else
56.         {
57.         len=strlen(op)-2;
58.         lc=lc+len;}
59.         }
60.         else if(strcmp(m1,"RESB")==0)
61.         {
62.         op1=atoi(op);
63.         lc=lc+op1;
64.         }
65.         fscanf(f1,"%s%s",la,m1);
66.         }
67.         if(strcmp(m1,"END")==0)
68.         {
69.         printf("Program length =\n%d",lc-sa);
70.         }
71.         fclose(f1);
72.         fclose(f3);
73.         getch();
74.         }
```

Input.txt		
COPY	START	1000
-	LDA	ALPHA
-	ADD	ONE
-	SUB	TWO
-	STA	BETA
ALPHA	BYTE	C'KLNCE
ONE	RESB	2
TWO	WORD	5
BETA	RESW	1
-	END	-

Scanned by CamScanner

Optab.txt	
LDA	00
STA	23
ADD	01
SUB	05

Output:

Symtab.txt	
1012	ALPHA
1017	ONE
1019	TWO
1022	BETA

COPY	START	1000	
1000	-	LDA	ALPHA
1003	-	ADD	ONE
1006	-	SUB	TWO
1009	-	STA	BETA
1012	ALPHA	BYTE	C'KLNCE
1017	ONE	RESB	2
1019	TWO	WORD	5
1022	BETA	RESW	1
1025	-	END	-
Program length = 25			

Scanned by CamScanner

8. C Program to implement Absolute Loader.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
FILE *fp;
int i,addr1,l,j,staddr1;
char name[10],line[50],name1[10],addr[10],rec[10],ch,staddr[10];
clrscr();
printf("enter program name:" );
scanf("%s",name);
fp=fopen("abssrc.txt","r");
fscanf(fp,"%s",line);
for(i=2,j=0;i<8,j<6;i++,j++)
name1[j]=line[i];
name1[j]='\0';
printf("name from obj. %s\n",name1);
if(strcmp(name,name1)==0)
{
do
{
fscanf(fp,"%s",line);
if(line[0]=='T')
{
for(i=2,j=0;i<8,j<6;i++,j++)
staddr[j]=line[i];
staddr[j]='\0';
staddr1=atoi(staddr);
i=12;
while(line[i]!='$')
{
if(line[i]!='^')
{
printf("00%d \t %c%c\n", staddr1,line[i],line[i+1]);
staddr1++;
i=i+2;
}
else i++;
}
}
else if(line[0]='E')
fclose(fp);
```


Scanned by CamScanner

```
44.         }while(!feof(fp));
45.         }
46.         getch();
47.         }
```

INPUT (ABSSRC.TXT)
H^SAMPLE^001000^0035
T^001000^0C^001003^071009S
T^002000^03^111111S
E^001000

OUTPUT
enter program name: SAMPLE
name from obj. SAMPLE
001000 00
001001 10
001002 03
001003 07
001004 10
001005 09
002000 11
002001 11
002002 11

Scanned by CamScanner

9.C program to find the FIRST in context free grammar.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

void main()
{

        char t[5],int[10],p[5][5],first[5][5],temp;
        int i,j,not,nont,k=0,f=0;
        clrscr();

        printf("\nEnter the no. of Non-terminals in the grammer:");
        scanf("%d",&nont);
        printf("\nEnter the Non-terminals in the grammer:\n");
        for(i=0;i<nont;i++)
        {
                scanf("\n%c",&nt[i]);
        }

        printf("\nEnter the no. of Terminals in the grammer: ( Enter { for absiline )
");
        scanf("%d",&not);
        printf("\nEnter the Terminals in the grammer:\n");
        for(i=0;i<not||t[i]=='S';i++)
        {
                scanf("\n%c",&t[i]);
        }

    for(i=0;i<nont;i++)
    {
            p[i][0]=nt[i];
            first[i][0]=nt[i];
    }

    printf("\nEnter the productions :\n");
    for(i=0;i<nont;i++)

         {
                scanf("%c",&temp);
                printf("\nEnter the production for %c ( End the production with 'S'
 sign ) :",p[i][0]);
                for(j=0;p[i][j]!='S';)
                {
                        j+=1;
                        scanf("%c",&p[i][j]);
                }
        }

        for(i=0;i<nont;i++)
        {
                printf("\nThe production for %c -> ",p[i][0]);
                for(j=1;p[i][j]!='S';j++)
                {
                        printf("%c",p[i][j]);
                }
        }
```

Scanned by CamScanner

```
for(i=0;i<nont;i++)
{
        f=0;
        for(j=1;p[i][j]!='$';j++)
        {
                for(k=0;k<not;k++)
                {
                        if(f==1)
                        break;

                        if(p[i][j]==t[k])
                        {
                                first[i][j]=t[k];
                                first[i][j+1]='$';
                                f=1;
                                break;
                        }

                        else if(p[i][j]==nt[k])


                          {
                                first[i][j]=first[k][j];
                                if(first[i][j]=='{')
                                continue;
                                first[i][j+1]='$';
                                f=1;
                                break;
                           }
                      }
                }
        }

        for(i=0;i<nont;i++)
        {
                printf("\n\nThe first of %c -> ",first[i][0]);
                for(j=1;first[i][j]!='$';j++)
                {
                        printf("%c\t",first[i][j]);
                }
        }

getch();
}
```

Scanned by CamScanner

10.C Program to implement Shift Reduce Parser for the given grammar

E →E+E
E→E*E
E→(E)
E → id

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
void main()
  {
     clrscr();
     puts("GRAMMAR is E->E+E \n E->E*E \n E->(E) \n E->id");
     puts("enter input string ");
     gets(a);
     c=strlen(a);
     strcpy(act,"SHIFT->");
     puts("stack \t input \t action");
     for(k=0,i=0; j<c; k++,i++,j++)
      {
        if(a[j]=='i' && a[j+1]=='d')
          {
             stk[i]=a[j];
             stk[i+1]=a[j+1];
             stk[i+2]='\0';
             a[j]=' ';
             a[j+1]=' ';
             printf("\n$%s\t%s$\t%sid",stk,a,act);
             check();
          }
        else
          {
             stk[i]=a[j];
             stk[i+1]='\0';
             a[j]=' ';
             printf("\n$%s\t%s$\t%ssymbols",stk,a,act);
             check();
          }
      }
     getch();
  }
void check()
  {
    strcpy(ac,"REDUCE TO E");
    for(z=0; z<c; z++)
      if(stk[z]=='i' && stk[z+1]=='d')
        {
          stk[z]='E';
```


Scanned by CamScanner

```
            stk[z+1]='\0';
            printf("\n$%s\t%s$\t%s",stk,a,ac);
            j++;
          }
       for(z=0; z<c; z++)
         if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')
          {
            stk[z]='E';
            stk[z+1]='\0';
            stk[z+2]='\0';
            printf("\n$%s\t%s$\t%s",stk,a,ac);
            i=i-2;
          }
       for(z=0; z<c; z++)
         if(stk[z]=='E' && stk[z+1]=='*' && stk[z+2]=='E')
          {
            stk[z]='E';
            stk[z+1]='\0';
            stk[z+1]='\0';
            printf("\n$%s\t%s$\t%s",stk,a,ac);
            i=i-2;
          }
      for(z=0; z<c; z++)
        if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')')
          {
            stk[z]='E';
            stk[z+1]='\0';
            stk[z+1]='\0';
            printf("\n$%s\t%s$\t%s",stk,a,ac);
            i=i-2;
          }
    }
```

```
GRAMMAR is E=E*E
E=E*E
E=(E)
E=id
input string is
(id*id)*id
stack                input                action

$(              id*id)*id$      SHIFT ->symbols
$(id              *id)*id$      SHIFT ->id
$(E               *id)*id$      REDUCE TO E
$(E*                id)*id$     SHIFT ->symbols
$(E*id                )*id$     SHIFT ->id
$(E*E                 )*id$     REDUCE TO E
$(E                   )*id$     REDUCE TO E
$(E)                    *id$    SHIFT ->symbols
$E                      *id$    REDUCE TO E
$E*                       id$   SHIFT ->symbols
$E*id                       $   SHIFT ->id
$E*E                        $   REDUCE TO E
$E                          $   REDUCE TO E_
```


Scanned by CamScanner