File Name: vi first.c

Program for Tokenization (counting the no of characters, lines, spaces, words, tabs, integer, float, Sum of the given integer & float etc..,).

Program:

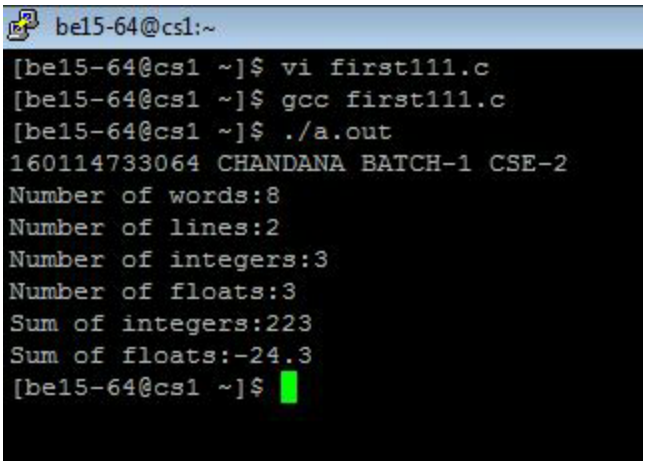
```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
static    int sumint;
   float sumfloat;
int main()
{
            FILE *fp;
            char s[100];
            char c;
            int sum=0,ws;
            int i=0,j=0,lines=0,words=0,integers=0,floats=0,check;
            fp = fopen("file.txt","r");
            while((c=fgetc(fp))!=EOF)
             {
                    if(c=='\n')
                                lines++;
                                if(c!=' '&&c!='\n')
                                        {
                                                s[i]=c;
                                            i++;
                                }

                                            if(c==' '||c=='\n')
                                          {
                                                    words++;
                                                        ws=i;
                                                        check=verify(s,i);
                                                        if(check==0)
                                                  integers++;
                                                     else if(check==2)
                                                          floats++;
                                                        i=0;
                                        }
                    }
                    printf("\nNumber of words:%d\n",words);
                printf("Number of lines:%d\n",lines);
                    printf("Number of integers:%d\n",integers);
                       printf("Number of floats:%d\n",floats);
                       printf("Sum of integers:%d\n",sumint);
                       printf("Sum of floats:%g\n",sumfloat);
}
int verify(char *c,int x)
{
                int i=0,flag=0,temp=0;
                     float temp1=0.0;
```

```
                    char a[100],a1[100];
                  for(i=0;i<x;i++)
                {
                                  if(((d[i])>=48&&(d[i])<=57)&&(d[i]!='+'||d[i]!='-'))
                                       flag++;
                }
                        i=0;
                if(d[i]=='+'||d[i]=='-')
                {
                                  flag=1;
                                   for(i=0;i<x;i++)
                                     {
                                                  if(((d[i])>=48&&(d[i])<=57)&&(d[i]!='+'||d[i]!='-'))
                                                          flag++;
                        }
              }
                  if(flag==x)
                    {
                            i=0;
                                 if(d[0]=='+'||d[0]=='-')
                                  {
                                                  for(i=1;i<x;i++)
                        {
                                       a[i-1]=d[i];
                        }
                        a[i-1]='\0';
                                                 temp=atoi(a);
                                        //printf("\n temp:%d",temp);
                                                  if(d[0]=='+')
                                                       {
                                        sumint=sumint+temp;
                                                                //     printf("sum of int in if
+:%d ",sumint);
                                                       }
                                           else if(d[0]=='-')
                                                       {
                                        sumint=sumint-temp;
                        //                      printf("sum of int in else -:%d",sumint);
                        }
                            }
                            else
                            {
                                        for(i=0;i<x;i++)
{
   a[i]=d[i];
// printf("\n in c without sign :%d",d[i]);
}
a[i]='\0';
                                        temp=atoi(a);
                                        //printf("\n in else temp:%d",temp);
                                        sumint=sumint+temp;
                            }
                            return 0;
              }
```

```
                else
            {
                        i=0;flag=0;int q=0;
                        for(i=0;i<x;i++)
                        {
                                    if(((d[i])>=48&&(d[i])<=57&&(d[i]!='+'||d[i]!='-')))
                                                flag++;
                                    if(d[i]=='.')
                                        {
            flag++;
            q++;
}
if(q>1)
break;

                        }
                        i=0;
                        if(d[0]=='+'||d[0]=='-')
                        {q=0;
                                    flag=1;
                                    for(i=0;i<x;i++)
                                    {
                                                if(((d[i])>=48&&(d[i])<=57&&(d[i]!='+'||d[i]!='-')))
                                                            flag++;
                                                                if(d[i]=='.')
                                                            {
   flag++;
   q++;
}
if(q>1)
break;
                                    }
                        }
                        if(flag==x)
                        {
                                    i=0;
                                    if(d[0]=='+'||d[0]=='-')
                                    {
                                                for(i=1;i<x;i++)
                                                            a1[i-1]=d[i];
                                                            a1[i-1]='\0';
                                                temp1=atof(a1);
                                        //      printf("\ntemp1 :%f",temp1);
                                                if(d[0]=='+')
                                                            sumfloat=sumfloat+temp1;
                                                else if(d[0]=='-')
                                                            sumfloat=sumfloat-temp1;
                                    }
                                    else
                                    {
                                        int p=0,m=0;
                                        for(i=0;i<x;i++)
                                        {
a1[i]=d[i];
```

```
//    printf(" \n in c without sign:%c",c[i]);
}
                                                a1[i]='\0';
                                                for(i=0;a1[i]!='.';i++)
                                                {
                                                p++;
                                                //    printf("\n p:%d",p);
}
for(m=0;p>0;m++)
{
        char sp=a1[m];
           sp=sp-'0';
        //    printf("\n %d",sp);
        temp1=temp1+(sp*pow(10,--p));
        //    printf("\n %f",temp1);
}
//    printf("\n%c",a1[m]);
p=0;
i++;
m++;
for(;a1[i]!='\0';i++)
{
        p++;
}
for(;p>0;m++)
{
        char np=a1[m];
        np=np-'0';
        //    printf("\n np:%d",np);
        temp1=temp1+(np*pow(10,-p));
        p--;
        //    printf("\n %f",temp1);
}
 //             printf("\nflag without sign float:%d\n",p);
                                                sumfloat=sumfloat+temp1;
                                        }

                                        return 2;
                                }
                                else
                                        return -1;
                }
}
```

Output:



File Name: vi scanner.c

Program to implement Scanner using C.

Program:

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
int main()
{
	FILE *input, *output;
	int l=1;
	int t=0;
	int j=0;
	int i,flag;
	char ch,str[20];
	input = fopen("input.txt","r");
	output = fopen("output.txt","w");
	char keyword[30][30] = {"int","main","if","else","do","while"};
	fprintf(output,"160114733064 CHANDANA CSE-2 BATCH - 1\n");
	fprintf(output,"Line no.	\t Token no.		\t Token		\t Lexeme\n\n");
```

```
        while(!feof(input))
        {
                i=0;
                flag=0;
                ch=fgetc(input);
                if( ch=='+' || ch== '-' || ch=='*' || ch=='/' )
                {
                        fprintf(output,"%7d\t\t %7d\t\t Operator\t %7c\n",l,t,ch);
                        t++;
                }
                if(ch=='h')
                {
                        fprintf(output,"%7d\t\t %7d\t\t Extension\t\t %7c\n",l,t,ch);
                        t++;
                }
                else if( ch==';' || ch=='{' || ch=='}' || ch=='(' || ch==')' || ch=='?' || ch=='@' || ch=='!' ||
ch=='%')
                {
                        fprintf(output,"%7d\t\t %7d\t\t Special symbol\t %7c\n",l,t,ch);
                        t++;
                }
                else if(isdigit(ch))
                {
                        fprintf(output,"%7d\t\t %7d\t\t Digit\t\t %7c\n",l,t,ch);
                        t++;
                }
                else if(isalpha(ch))
                {

                        str[i]=ch;
                        i++;
                        ch=fgetc(input);
                        while(isalnum(ch) && ch!=' ')
                        {
                                str[i]=ch;
                                i++;
                                ch=fgetc(input);
                        }
                        str[i]='\0';
                        for(j=0;j<=5;j++)
                       {
                                if(strcmp(str,keyword[j])==0)
                                {
                                        flag=1;
                                        break;
                                }
                        }
                        if(flag==1)
                        {
                                fprintf(output,"%7d\t\t %7d\t\t Keyword\t %7s\n",l,t,str);
                                t++;
                        }
                        else
                        {
```

```
                        fprintf(output,"%7d\t\t %7d\t\t Identifier\t %7s\n",l,t,str);
                        t++;
                }

        }
        else if(ch=='\n')
        {
                l++;
        }
    }
    fclose(input);
    fclose(output);
    return 0;
}
```

Output:

```
be15-64@cs1:~
160114733064 CHANDANA CSE-2 BATCH - 1
Line no.        Token no.              Token          Lexeme

      1               0               Identifier     include
      1               1               Identifier       stdio
      1               2               Extension                  h
      2               3               Identifier     include
      2               4               Identifier       ctype
      2               5               Extension                  h
      3               6               Identifier     include
      3               7               Identifier      string
      3               8               Extension                  h
      4               9               Keyword            int
      4              10               Keyword           main
      4              11               Special symbol       )
      5              12               Special symbol       {
      6              13               Identifier        FILE
      6              14               Operator             *
      6              15               Identifier       input
      6              16               Operator             *
      6              17               Identifier      output
      7              18               Keyword            int
      7              19               Identifier           l
      7              20               Digit                1
      7              21               Special symbol       ;
      8              22               Keyword            int
      8              23               Identifier           t
      8              24               Digit                0
      8              25               Special symbol       ;
      9              26               Keyword            int
      9              27               Identifier           j
      9              28               Digit                0
      9              29               Special symbol       ;
     10              30               Keyword            int
     10              31               Identifier           i
     10              32               Identifier       flag
     11              33               Identifier       char
     11              34               Identifier         ch
     11              35               Identifier        str
     11              36               Digit                2
     11              37               Digit                0
     11              38               Special symbol       ;
     12              39               Identifier       input
-- INSERT --
```

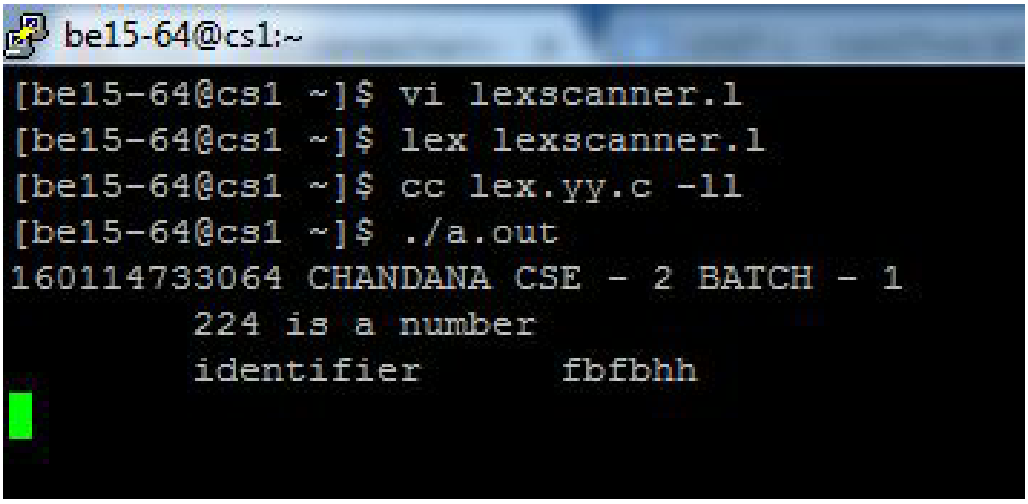
File Name: vi lexscanner.l

Program to implement Scanner application using LEXtool.

Program:

```
%{

/* LEXprogram for standalone scanner*/

int COMMENT=0;

%}

id [a-z][a-z0-9]*

%%

#.* {printf("\n%s is a preprocessor directive",yytext);}
int|double|char|float {printf("\n%s is a keyword",yytext);}
if|then|endif|else {printf("\n%s is a keyword",yytext);}
"/*" {COMMENT=1;}
"*/" {COMMENT=0;}
{id}\( {if(!COMMENT)printf("\n\nFUNCTION\n\t%s",yytext);}
{id}(\[[0-9]*\])? {if(!COMMENT)printf("\n\tidentifier\t%s",yytext);}
\{ {if(!COMMENT)printf("\n\tblock begins");ECHO;}
\} {if(!COMMENT)printf("\n\tblock ends");ECHO;}
\".*\" {if(!COMMENT)printf("\n\t %s is a tring",yytext);}
[+\-]?[0-9]+ {if(!COMMENT)printf("\n\t%s is a number",yytext);}
\( {if(!COMMENT) printf("\n\t");ECHO;printf("\t delimiter open paranthesis\n");}
\) {if(!COMMENT) printf("\n\t");ECHO;printf("\t delimiter close paranthesis\n");}
\; {if(!COMMENT) printf("\n\t");ECHO;printf("\n\t delim semicolon");}
\= {if(!COMMENT) printf("\n\t%s is an assignment operator");}
\<|\< printf("\n\t%s is relational operator");
"+"|"-"|"*"|"/" {printf("\n\t is a binary operator\n" );}
"\n" ;

%%

main()
{
	printf("160114733064 CHANDANA CSE - 2 BATCH - 1");
	yyin=fopen("one.txt","r");
	yylex();
	printf("\n");
}
int yywrap()
{
	return 0;
}
```

Output:



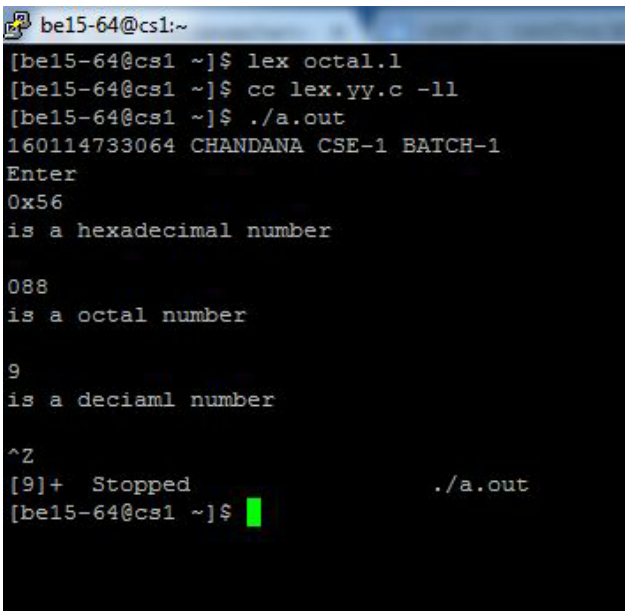
File Name: vi octal.l

Program to identify the Octal or Hexadecimal number using lex tool.

```
%{
#include<stdio.h>
%}

Oct [0][0-9]+
Hex [0][X|x][0-9 A-F a-f]+
Dec [1-9]*
```

```
Inv [[+-?][0-9]*|[0-9][+-]*]
%%
{Hex} {printf("is a hexadecimal number \n");}
{Oct} {printf("is a octal number \n");}
{Dec} {printf("is a deciaml number \n");}
{Inv} {printf("Invalid");}
%%
int yywrap()
{
return 1;
}
main()
{
printf("160114733064 CHANDANA CSE-1 BATCH-1\n");
printf("Enter \n");
yylex();
}
```

Output:



File Name: vi upper.l

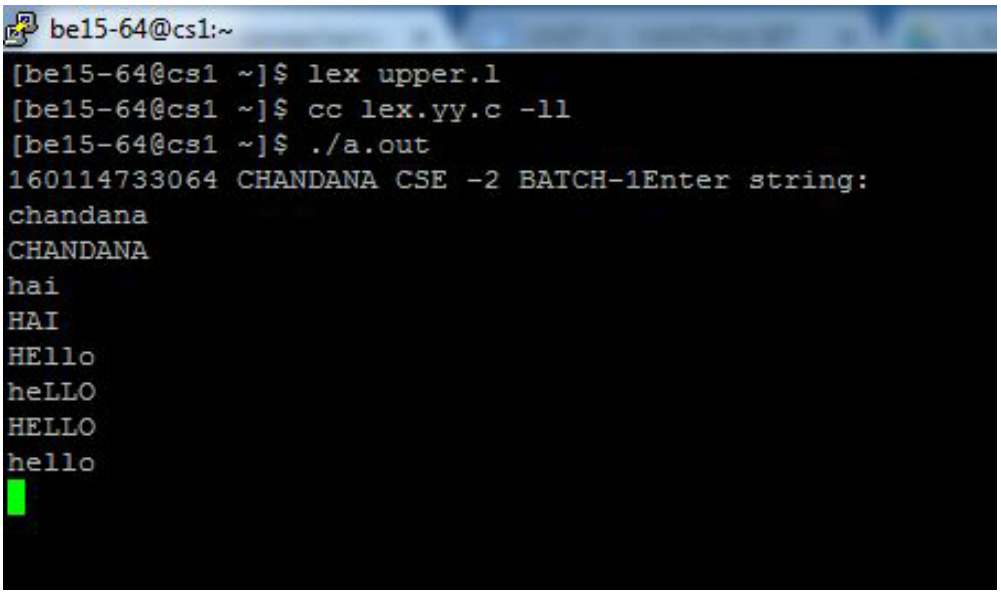
Program to capitalize the input string using LEX

Program:

```
%{
#include<stdio.h>
#include<string.h>
char ch[10];
%}
lower[a-z]
upper[A-Z]
```

```
%%
{lower} {printf("%c",(yytext[0]-32));}
{upper} {printf("%c",(yytext[0]+32));}
%%
int yywrap()
{
return 1;
}
main()
{
printf("160114733064 CHANDANA CSE -2 BATCH-1");
printf("Enter string: \n");
yylex();
}
```

Output:



File Name: vi precision.l

Program: Program to find real precision using lex

```
%{
#include<stdio.h>
#include<string.h>
int f,i,j;
%}

%%
[+-]?[0-9]+ {printf("\n%s is an integer!",yytext);}
[+-]?[0-9]*[.][0-9]+ {f=0; for(i=0;i<yyleng;i++)
{if(yytext[i]=='.') { j=i+1; break;} }
   for(;j<yyleng;j++)
f++;
   printf("\n%s is a floating number with a precision of %d!",yytext,f);}
```

```
[0-9a-zA-Z]+[.][0-9+-.a-zA-Z]+ {printf("\ninvalid!!!");}
[\n]     {return 0;}
%%

int main()
{
printf("Enter a number :\n");
yylex();
}

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

Output:

File Name: vi vowels.l

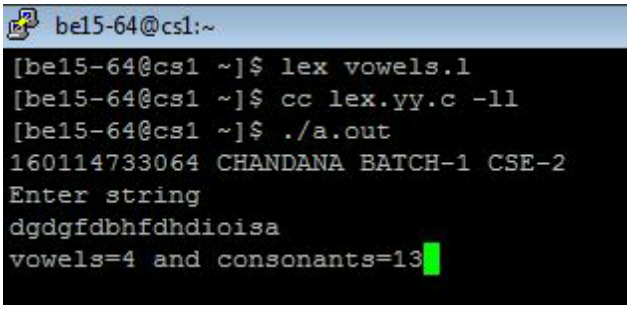
Program to count the number of vowels and consonants in a given string using Lex.

Program:

```
%{
#include<stdio.h>
int vowel=0,cons=0;
%}
%%
"a"|"e"|"i"|"o"|"u"|"A"|"E"|"I"|"O"|"U" {vowel++;}
[a-zA-Z] {cons++;}
[\n] {printf("vowels=%d and consonants=%d",vowel,cons);
    }
%%
int yywrap()
{
return 1;
}
main()
{
```

```
printf("Shriya\n160114733078\nEnter string \n");
yylex();
}
```

Output:



File Name: vi numbers.c

Program to count no of: a) +ve and –ve integers b) +ve and –ve fractions

Program:

```
%{
   int postiveno=0;
   int negtiveno=0;
   int positivefractions=0;
   int negativefractions=0;
%}

DIGIT [0-9]
%%

\+?{DIGIT}+                    postiveno++;
-{DIGIT}+                      negtiveno++;

\+?{DIGIT}*\.{DIGIT}+          positivefractions++;
-{DIGIT}*\.{DIGIT}+            negativefractions++;
.;
%%

main()
{
   yylex();
printf("160114733064 CHANDANA BATCH-1 CSE-2 \n");
```

```
    printf("\nNo. of positive numbers: %d",postiveno);
    printf("\nNo. of Negative numbers: %d",negtiveno);
    printf("\nNo. of Positive fractions: %d",positivefractions);
    printf("\nNo. of Negative fractions: %d\n",negativefractions);
}
```

Output:

```
be15-64@cs1:~
+12,-123,1.1,-1.1,12,-2,-3,2.1,3.2,5.1,-5.5,-6.1,-7.7,-8.8
~
```

```
be15-64@cs1:~
[be15-64@cs1 ~]$ lex fraction.l
[be15-64@cs1 ~]$ cc lex.yy.c -ll
[be15-64@cs1 ~]$ ./a.out <a.txt

160114733064 CHANDANA BATCH-1 CSE-2

No. of positive numbers: 2
No. of Negative numbers: 3
No. of Positive fractions: 4
No. of Negative fractions: 5
[be15-64@cs1 ~]$
```

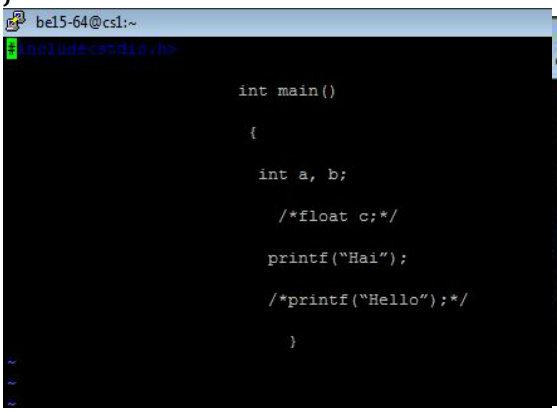
File Name: vi comment.l

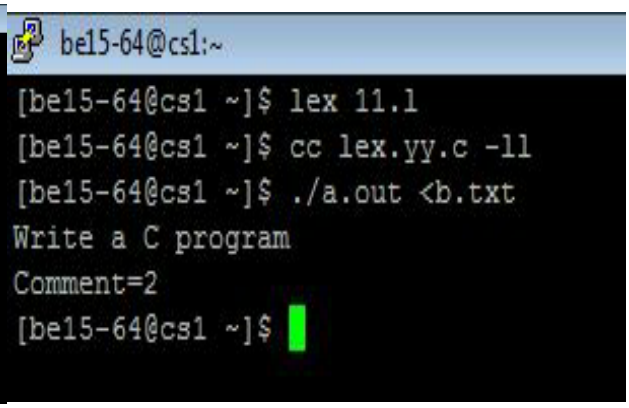
Program to count the no of comment line in a given C program. Also eliminate them and copy that program into separate file using Lex.

Program:

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

%%
"printf" {fprintf(yyout,"writef"); pfc++;}
"scanf" {fprintf(yyout,"readf"); sfc++;}
%%

main(int argc, char *argv[])
{
printf("160114733064 CHANDANA BATCH-1 CSE-2 \n");
if(argc!=3)
{
printf("Usage: ./a.out in.txt out.txt\n");
exit(0);
}
yyin=fopen(argv[1],"r");
yyout=fopen(argv[2],"w");
yylex();
printf("\n the number of printf lines = %d\n",pfc);
printf("\n the number of scanf lines = %d\n",sfc);
}
```

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





File Name: vi writeread.l

Program to count the no of 'scanf' and 'printf' statements in a C program. Replace them with 'readf' and 'writef' statements respectively using lex.
Program:

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

%%
"printf" {fprintf(yyout,"writef"); pfc++;}
"scanf" {fprintf(yyout,"readf"); sfc++;}
%%

main(int argc, char *argv[])
{
if(argc!=3)
{
printf("Usage: ./a.out in.txt out.txt\n");
exit(0);
}
yyin=fopen(argv[1],"r");
yyout=fopen(argv[2],"w");
yylex();
printf("\n the number of printf lines = %d\n",pfc);
printf("\n the number of scanf lines = %d\n",sfc);
}

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

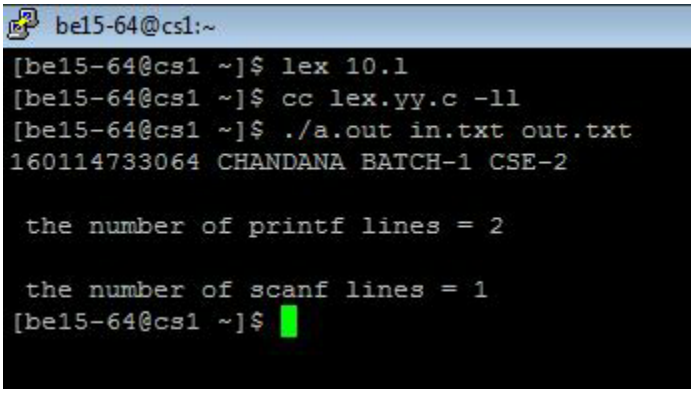
CONTENTS OF IN.TXT FILE

```
#include<stdio.h>
int main()
{
int a,b,c;
printf("enter two numbers");
scanf("%d %d",&a,&b);
c=a+b;
printf("sum is %d",c);
return 0;
}
```

CONTENTS OF OUT.TXT FILE

```
#include<stdio.h>
int main()
{
int a,b,c;
writef("enter two numbers");
readf("%d %d",&a,&b);
c=a+b;
writef("sum is %d",c);
return 0;
}
```

Output:



File Name: vi rdp.c

Program to implement Recursive descent parser for the given grammar

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

char input[10];
int i,error;
void E();
void T();
void Eprime();
void Tprime();
void F();
main()
{
i=0;
error=0;
printf("Enter an arithmetic expression  :  "); // Eg: a+a*a
gets(input);
E();
if(strlen(input)==i&&error==0)
printf("\nAccepted..!!!\n");
else printf("\nRejected..!!!\n");
}


void E()
{
T();
Eprime();
}
void Eprime()
{
if(input[i]=='+')
{
i++;
T();
Eprime();
}
}
void T()
{
F();
```

```
        Tprime();
}
void Tprime()
{
        if(input[i]=='*')
        {
                                        i++;
                                        F();
                                        Tprime();
                                        }
                                        }
        void F()
        {
                if(isalnum(input[i]))i++;
                else if(input[i]=='(')
                {
                i++;
                E();
                if(input[i]==')')
                i++;

                else error=1;
                    }

                else error=1;
                }
```

```
be15-64@cs1:~
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter an arithmetic expression   :

Rejected..!!!
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter an arithmetic expression   :  a+a(a*a)

Rejected..!!!
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter an arithmetic expression   :  a

Accepted..!!!
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter an arithmetic expression   :  a+a+a*a

Accepted..!!!
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter an arithmetic expression   :  ++a

Rejected..!!!
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter an arithmetic expression   :  a***a

Rejected..!!!
[be15-64@cs1 ~]$
```

Program to find FIRST elements for the given grammar.

```
#include<stdio.h>
#include<ctype.h>
int main()
{
```

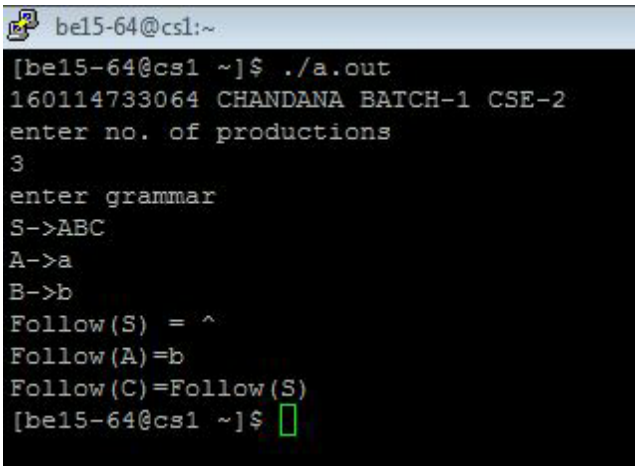
```
        int i,n,j,k;
        char str[10][10],f;
        printf("Enter the number of productions\n");
        scanf("%d",&n);
        printf("Enter grammar\n");
        for(i=0;i<n;i++)
            scanf("%s",&str[i]);
        for(i=0;i<n;i++)
        {
            f= str[i][0];
            int temp=i;
              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;
        }
}
```

```
be15-64@cs1:~
[be15-64@cs1 ~]$ vi 12.c
[be15-64@cs1 ~]$ gcc 12.c
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter the number of productions
3
Enter grammar
S->AB
A->a
B->b
First(S)=a
First(A)=a
First(B)=b
[be15-64@cs1 ~]$
```

Program to find FOLLOW elements for the given grammar.

```
#include<stdio.h>

main()
{
        int np,i,j,k;
        char prods[10][10],follow[10][10],Imad[10][10];
        printf("enter no. of productions\n");
        scanf("%d",&np);
        printf("enter grammar\n");
        for(i=0;i<np;i++)
        {
           scanf("%s",&prods[i]);
        }

        for(i=0; i<np; i++)
        {
        if(i==0)
                {
                 printf("Follow(%c) = ^\n",prods[0][0]);//Rule1
                }
                for(j=3;prods[i][j]!='\0';j++)
                {
                int temp2=j;
                //Rule-2: production A->xBb then everything in first(b) is in follow(B)
                if(prods[i][j] >= 'A' && prods[i][j] <= 'Z')
                {
                 if((strlen(prods[i])-1)==j)
                {
                        printf("Follow(%c)=Follow(%c)\n",prods[i][j],prods[i][0]);
                 }
                int temp=i;
                char f=prods[i][j];
                if(!isupper(prods[i][j+1])&&(prods[i][j+1]!='\0'))
```

```
                printf("Follow(%c)=%c\n",f,prods[i][j+1]);
                if(isupper(prods[i][j+1]))
                {
                                repeat:
                        for(k=0;k<np;k++)
                        {
                        if(prods[k][0]==prods[i][j+1])
                        {
                                if(!isupper(prods[k][3]))
                                {
                        printf("Follow(%c)=%c\n",f,prods[k][3]);
                                                }
                        else
                                {
                                i=k;
                                j=2;
                                goto repeat;
                                }
                        }
                                }
                                }
                                i=temp;
                        }
                        j=temp2;
                }
        }
}
```



Program to implement calculator using Yacc tool.

Calc.l

```
%{
        #include<stdio.h>
```

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

%%

[0-9]+  {yylval.dval = atoi( yytext ); return DIGIT;}
\n|.    return yytext[0];
%%
Calc.y
%{
 /*E->E+E|E*E|(E)|DIGIT*/
#include<stdio.h>
%}

%union
{
        int dval;
}

%token  <dval> DIGIT
%type   <dval> expr
%type   <dval> expr1

%%

line    :       expr '\n'                       {printf("%d\n",$1);}
        ;
expr    :       expr '+' expr1                  {$$ = $1  + $3 ;}
        |       expr '-' expr1                  {$$ = $1  - $3 ;}
```

```
        |       expr '*' expr1                {$$ = $1 * $3 ;}
        |       expr '/' expr1                {$$ = $1  /  $3 ;}
        |       expr1
        ;

expr1   :       '('expr')'                  {$$=$2;}
        |       DIGIT
        ;
%%


int main()
{
        printf("160114733064 CHANDANA CSE - 2 BATCH - 1\n");
        yyparse();
}
yyerror(char *s)
{
        printf("%s",s);
}
```

```
[be15-64@cs1 ~]$ lex calc.l
[be15-64@cs1 ~]$ yacc -d calc.y
[be15-64@cs1 ~]$ gcc lex.yy.c y.tab.c -ll
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA CSE - 2 BATCH - 1
2+2
4
^Z
[4]+  Stopped                 ./a.out
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA CSE - 2 BATCH - 1
4/2
2
^Z
[5]+  Stopped                 ./a.out
[be15-64@cs1 ~]$
```

Program to recognize strings ‘aaab’, ‘abbb’, ‘ab’ and ‘a’ using grammar (a nb n , n>=0)

18.l

```
%{
#include "y.tab.h"
%}
%%
a return A;
b return B;
.|\n return yytext[0];
%%
```

18.y

```
%{
#include<stdio.h>
int valid;
%}
%token A B
%%
str:S\n' {valid=1;return 0;}
SA SB
```

```
 |
 ;
%%
main()
{
printf("160114733064 CHANDANA BATCH-1 CSE-2\n");
 printf("Enter the string:\n");
   yyparse();
   if(valid==1)
   printf("\nvalid string");
}
yyerror(char *s)
{
        printf("%s",s);
}
```

```
be15-64@cs1:~
[be15-64@cs1 ~]$ vi 18.y
[be15-64@cs1 ~]$ vi 18.l
[be15-64@cs1 ~]$ lex 18.l
[be15-64@cs1 ~]$ yacc -d 18.y
[be15-64@cs1 ~]$ gcc lex.yy.c y.tab.c -ll
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter the string:
aabb

valid string[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter the string:
AABB
syntax error[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter the string:
AAAAAAA*B
syntax error[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter the string:
aaaaa*b
syntax error[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter the string:
a*b
syntax error[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter the string:
aaaaaaaaaaabbbbbbbbbbbbbb
syntax error[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH-1 CSE-2
Enter the string:
aabb

valid string[be15-64@cs1 ~]$
```

. Program to construct Goto

```
#include<stdio.h>


int main()
{
```

```
    const int maxInput = 5;
    int i;
    double number, average, sum=0.0;
printf("160114733064 CHANDANA BATCH – 1 CSE – 2\n");
    for(i=1; i<=maxInput; ++i)
    {
        printf("%d. Enter a number: ", i);
        scanf("%lf",&number);

    // If user enters negative number, flow of program moves to label jump
            if(number < 0.0)
                    goto jump;

                        sum += number; // sum = sum+number;
                          }

                            jump:

                              average=sum/(i-1);
                                printf("Sum = %.2f\n", sum);
                                  printf("Average = %.2f", average);

                                    return 0;
                                    }
```

```
be15-64@cs1:~
[be15-64@cs1 ~]$ gcc goto.c
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH - 1 CSE - 2
1. Enter a number: 123
2. Enter a number: 345
3. Enter a number: 567
4. Enter a number: 789
5. Enter a number: 90
Sum = 1914.00
Average = 382.80[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH - 1 CSE - 2
1. Enter a number: -456
Sum = 0.00
Average = -nan[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH - 1 CSE - 2
1. Enter a number: 00000
2. Enter a number: 0
3. Enter a number: 0
4. Enter a number: 8
5. Enter a number: 0
Sum = 8.00
[be15-64@cs1 ~]$
```

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

17.y

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

%token IF RELOP SNUMBER ID NL

%%
stmt: if_stmt NL {printf("No. of nested if statements=%d\n",count);exit(0);}
;
if_stmt : IF'('cond')''{'if_stmt'}' {count++;}
        |S
;
```

```
cond: x RELOP x
;
x:ID | NUMBER
;
%%

int yyerror(char *msg)
{
printf("the statement is invalid\n");
exit(0);
}

main()
{
Printf("160114733064 CHANDANA BATCH -1 CSE – 2\n");
printf("enter the statement\n");
yyparse();
}
```

17.l

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

%%
"if" {return IF;}
[sS][0-9]* {return S;}
"<"|">"|"=="|"<="|">="|"!=" {return RELOP;}
[0-9]+ {return NUMBER;}
[a-z][a-zA-Z0-9_]* {return ID;}
```

```
\n {return NL;}

. {return yytext[0];}

%%
```

```
be15-64@cs1:~
[be15-64@cs1 ~]$ vi 17.y
[be15-64@cs1 ~]$ vi 17.l
[be15-64@cs1 ~]$ yacc -d 17.y
[be15-64@cs1 ~]$ lex 17.l
[be15-64@cs1 ~]$ cc y.tab.c lex.yy.c  -ll
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH -1 CSE - 2
enter the statement
if(a<b){s}
No. of nested if statements=1
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH -1 CSE - 2
enter the statement
if(a<b){if(a>b){s}}
No. of nested if statements=2
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH -1 CSE - 2
enter the statement
if(a<b){if(a<b){if(a>b){s}}}
No. of nested if statements=3
[be15-64@cs1 ~]$
```

Program to construct predictive LL1 parsing table.

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
int isPresent(char c,char *s)
{
	int i;
	for(i=0;s[i]!='\0';i++)
	{
		if(s[i]==c)
			return 1;
	}
	 return 0;
}
int main()
{
	char grammar[20][20];
	char first[10][10];
	char follow[10][10];
	int i,j,k,n,te;char c;char t[10];
	printf("enter the number of productions: ");
	scanf("%d",&n);
```

```
        for(i=0;i<n;i++)
        {
                printf("enter the production");
                scanf("%s",&grammar[i]);
        }
        printf("enter the first for");
        for(i=0;i<n;i++)
        {
                printf("%c",grammar[i][0]);
                scanf("%s",&first[i]);
        }
        printf("enter the follow   for");
        for(i=0;i<n;i++)
{
                printf("%c",grammar[i][0]);
                scanf("%s",&follow[i]);
        }
        printf("enter the number of terminals");
        scanf("%d",&te);
        printf("Enter the terminals list");
        scanf("%s",&t);
        printf("\n-----------------------------------------");
        printf("\nNT\t");
        for(i=0;i<te;i++)
                printf("%c\t",t[i]);
        printf("$\t");
        for(i=0;i<n;i++)
        {
                printf("\n-----------------------------------------\n");
                printf("%c\t",grammar[i][0]);
                for(j=0;j<te;j++)
                {
                        if(isPresent(t[j],first[i])==1)
                        {
                                printf("%s\t",grammar[i]);
                        }
                        printf("\t");
                }

                if(isPresent('@',first[i])==1)
                {

                        for(j=0;j<te;j++)
                        {
                        if(isPresent(t[j],follow[i])==1)
                        {
                                printf("%s\t",grammar[i]);
                        }
                        printf("\t");
                        }

                }
                        printf("\t");
```

```
    }
    return 0;
}
```

be15-64@cs1:~

```
[be15-64@cs1 ~]$ vi LL1.c
[be15-64@cs1 ~]$ gcc LL1.c
[be15-64@cs1 ~]$ ./a.out
160114733064 CHANDANA BATCH - 1 CSE - 2
enter the number of productions : 3
enter the production S->AB
enter the production A->a
enter the production B->b
enter the first forS a
Aa
Bb
enter the follow  forS #
Ab
B#
enter the number of terminals 2
Enter the terminals list ab

-------------------------------------------
NT      a       b       #
-------------------------------------------
S       S->AB
-------------------------------------------
A       A->a
-------------------------------------------
[be15-64@cs1 ~]$
```