School of Engineering

List of Experiments

Sem-7IT **Subject: System Programming**

NO	TITLE
1	Write a program to implement the LexicalAnalyzer.
2	WAP using LEX to count the number of characters, words, spaces and lines in a given Input file.
3	WAP using LEX to count the numbers of Comment lines in a given C program.
4	WAP using LEX to recognize a valid arithmeticexpression and to recognize the identifiers And operators present. Print them separately.
5	WAP using LEX to recognize and count the Number of identifiers in a given input file.
6	WAP using YACC to recognize a valid arithmetic expression that uses operators +, – , * and /.
7	WAP using YACC to recognize a valid variable, Which starts with a letter, followed by anynumber of letters or digits?
8	Write a program to left factor the givengrammar.
9	Write a program to remove the Left Recursion from a given grammar.
10	Implement Recursive Descendent Parsing forthe given Grammar. E -> T + E / TT -> F * T / F F -> (E) / i.
11	Implement Predictive Parser for the givengrammar. E -> T + E / T T -> F * T / F F -> (E) / i.

School of Engineering

List of Experiments

Sem-7IT **Subject: System Programming**

12	Write a SAL program in text file and generate SYMTAB and LITTAB.			
13	Use macro features of C language.			
14	Write a program which generates Quadruple Table for the given postfix String.			
15	Write a C program to parse a given string using Predictive parsing for given Grammar. type ? simple | ?id | array [simple] of type simple ? integer | char | num dotdot num.			

1. Write a program to implement the Lexical Analyzer.

```
%{
#include<conio.h>
#include<stdio.h>
int COMMENT=0;
%}
identifier [a-zA-Z][a-zA-Z0-9]*
%%
#.* {printf("\n%s is a preprocessor directive",yytext);}
int |
float |
char |
double |
while |
for |
struct |
```

```
typedef |

do |

if |

break |

continue |

void |

switch |

return |

else |

goto {printf("\n\t%s is a keyword",yytext);}

"/*" {COMMENT=1;}{printf("\n\t %s is a COMMENT",yytext);}

{identifier}\( {if(!COMMENT)printf("\nFUNCTION
\n\t%s",yytext);}

\{  {if(!COMMENT)printf("\n BLOCK BEGINS");}

\}  {if(!COMMENT)printf("BLOCK ENDS ");}

{identifier}(\[[0-9]*\])? {if(!COMMENT) printf("\n %s
IDENTIFIER",yytext);}

\".*\" {if(!COMMENT)printf("\n\t %s is a STRING",yytext);}

[0-9]+ {if(!COMMENT) printf("\n %s is a NUMBER ",yytext);}
```

```
\)(\:)? {if(!COMMENT)printf("\n\t");ECHO;printf("\n");}

\( ECHO;

= {if(!COMMENT)printf("\n\t %s is an ASSIGNMENT
OPERATOR",yytext);}

\<= |

\>= |

\< |

== |

\> {if(!COMMENT) printf("\n\t%s is a RELATIONAL
OPERATOR",yytext);}

%%

int main(int argc, char **argv)

{

FILE *file;

file=fopen("var.c","r");

if(!file)

{

printf("could not open the file");

exit(0);
```

```
}
yyin=file;
yylex();
printf("\n");
return(0);
getch();
}
int yywrap()
{
return(1);
}
```

var.c

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
```

```
a=1;

b=2;

c=a+b;

printf("Sum:%d",c);

}
```

```
D:\5th sem\SP\Lex Programs\PR-1>Lex_Analyzer.exe

#include<stdio.h> is a preprocessor directive

#include<conio.h> is a preprocessor directive

        void is a keyword
FUNCTION
        main(
        )


 BLOCK BEGINS

        int is a keyword
 a IDENTIFIER,
 b IDENTIFIER,
 c IDENTIFIER;

 a IDENTIFIER
         = is an ASSIGNMENT OPERATOR
 1 is a NUMBER ;

 b IDENTIFIER
         = is an ASSIGNMENT OPERATOR
 2 is a NUMBER ;

 c IDENTIFIER
         = is an ASSIGNMENT OPERATOR
 a IDENTIFIER+
 b IDENTIFIER;

FUNCTION
        printf(
         "Sum:%d" is a STRING,
 c IDENTIFIER
        )
;
BLOCK ENDS

D:\5th sem\SP\Lex Programs\PR-1>
```

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

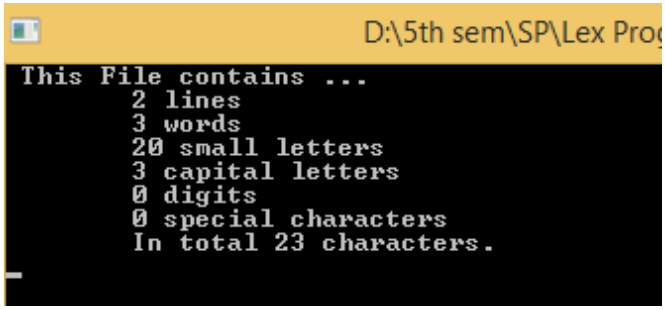
```
%{
#include<stdio.h>
#include<conio.h>
int lines=0, words=0,s_letters=0,c_letters=0, num=0,
spl_char=0,total=0;
%}
%%



\n { lines++; words++;}
[\t ' '] words++;
[A-Z] c_letters++;
[a-z]  s_letters++;
[0-9] num++;
. spl_char++;
%%
main(void)
{
yyin= fopen("data.txt","r");
yylex();
total=s_letters+c_letters+num+spl_char;
```

```
printf(" This File contains ...");
printf("\n\t%d lines", lines);
printf("\n\t%d words",words);
printf("\n\t%d small letters", s_letters);
printf("\n\t%d capital letters",c_letters);
printf("\n\t%d digits", num);
printf("\n\t%d special characters",spl_char);
printf("\n\tIn total %d characters.\n",total);
getch();
}

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

data.txt

December

August

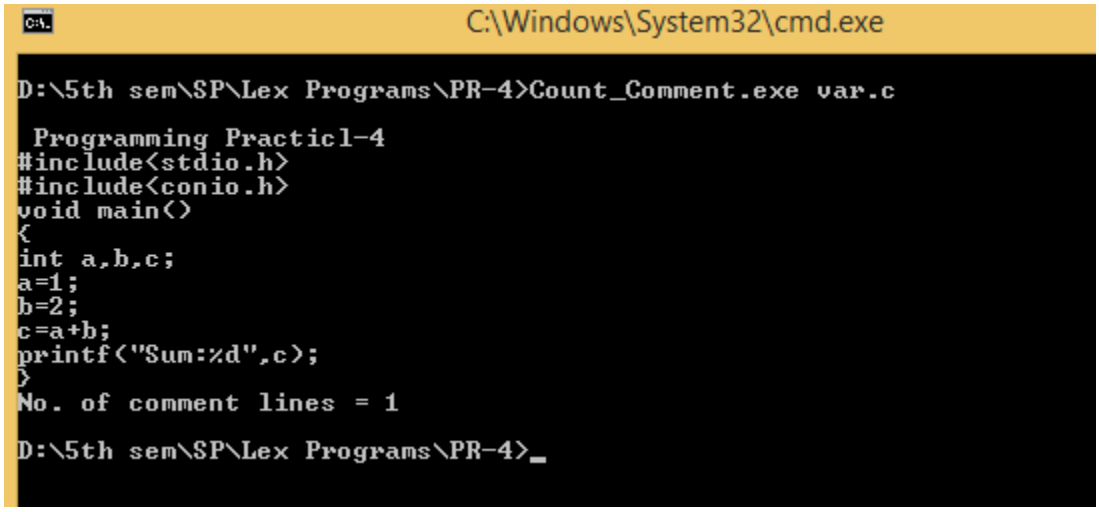


3. WAP using LEX to count the numbers of comment lines in a given C program.

```
%{
#include<stdio.h>
int c=0;
%}
%%
[/][/][a-zA-Z0-9]* {c++;}
[/][*][a-zA-Z0-9][*][/] {c++;}
%%
main(int argc,char*argv[ ])
{
yyin=fopen(argv[1],"r");
yyout=fopen(argv[2],"w");
yylex();
printf("\nNo. of comment lines = %d\n",c);
}
int yywrap()
{
return(1);
}
```

var.c

```
//System Programming Practicl-4
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
a=1;
b=2;
c=a+b;
printf("Sum:%d",c);
}
```



4. WAP using LEX to recognize a valid arithmetic expression and to recognize the identifiers and operators present. Print them separately.

```
%{
#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("\nValid expression\n");
```

```
}
else
{
printf("\nInvalid 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;
}
int yywrap()
{
return(1);
}
```

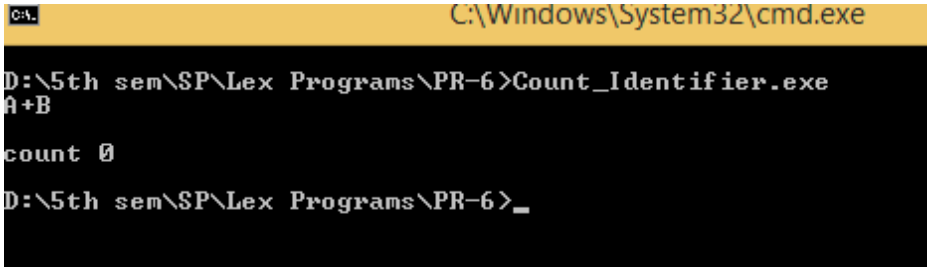
C:\Windows\System32\cmd.exe

```
D:\5th sem\SP\Lex Programs\PR-7>Arithmetic_Expression.exe
Enter the expression
1+2+3
123

Valid expression

Add=2
Sub=0
Mul=0
Div=0
^C
D:\5th sem\SP\Lex Programs\PR-7>_
```

5. WAP using LEX to recognize and count the number of identifiers in a given input file.

```
%{
#include<stdio.h>
int c=0;
char ch;
%}
%%
"int"|"float"|"double"|"long"|"short" {while(1)
{
ch=input();
if(ch==',')
c++;
else
if(ch==';')
c++;
break;
};}
. {;}
[\n] {;}
[a-z]+ {;}
%%
int main(int argc,char *argv[])
{
yyin=fopen(argv[1],"r");
```

```
yylex();
printf("\ncount %d\n",c);
fclose(yyin);
}
int yywrap()
{
return (1);
}
```



6. WAP using YACC to recognize a valid arithmetic expression that uses operators +, – , * and /.

```
%{
#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("\nValid expression\n");
```

```
}
else
{
printf("\nInvalid 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;
}
int yywrap()
{
return(1);
}
```

C:\Windows\System32\cmd.exe

```
D:\5th sem\SP\Lex Programs\PR-7>Arithmetic_Expression.exe
Enter the expression
1+2+3
123

Valid expression

Add=2
Sub=0
Mul=0
Div=0
^C
D:\5th sem\SP\Lex Programs\PR-7>_
```

7. WAP using YACC to recognize a valid variable, which starts with a letter, followed by any number of letters or digits.

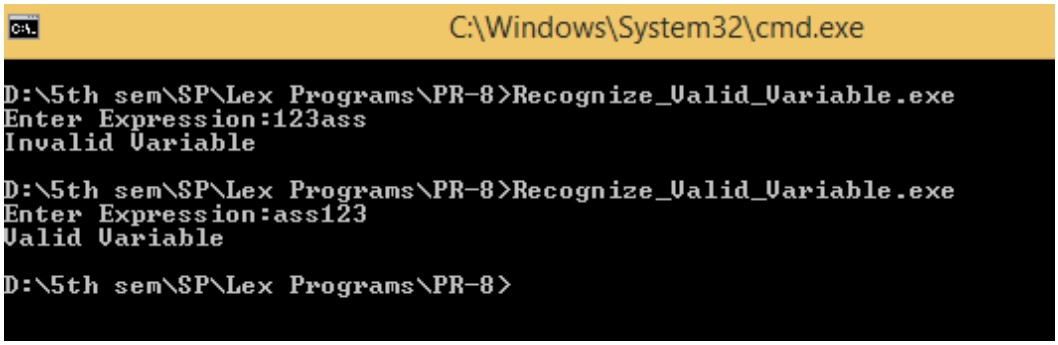
pr8.l

```
%{
#include "y.tab.h"
%}
%%
[a-zA-Z] {return ALPHA;}
[0-9]+ {return NUMBER;}
"\n" { return ENTER;}
%%
int yywrap()
{
return (1);
}
```

y.yacc

```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token ALPHA NUMBER ENTER ER
%%
```

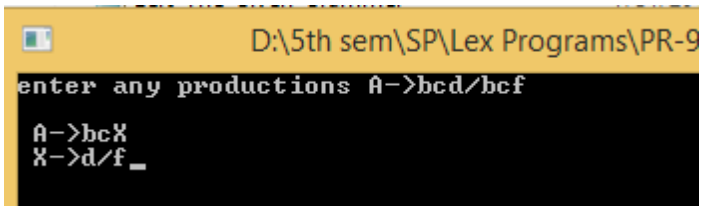
```
var:v ENTER {printf("Valid Variable\n");exit(0);}
    ;
v:ALPHA exp1
exp1:ALPHA exp1
|NUMBER exp1
|;
%%
yyerror()
{
printf("Invalid Variable\n");
}
void main(){
printf("Enter Expression:");
yyparse();
}
```



8. Write a program to left factor the given grammar.

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

void main()
{
char a[10],a1[10],a2[10],a3[10],a4[10],a5[10];
int i,j=0,k,l;



printf("enter any productions A->");

gets(a);

for(i=0;a[i]!='/';i++,j++)
a1[j]=a[i];
a1[j]='\0';

for(j=++i,i=0;a[j]!='\0';j++,i++)
a2[i]=a[j];
a2[i]='\0';
k=0;
l=0;
```

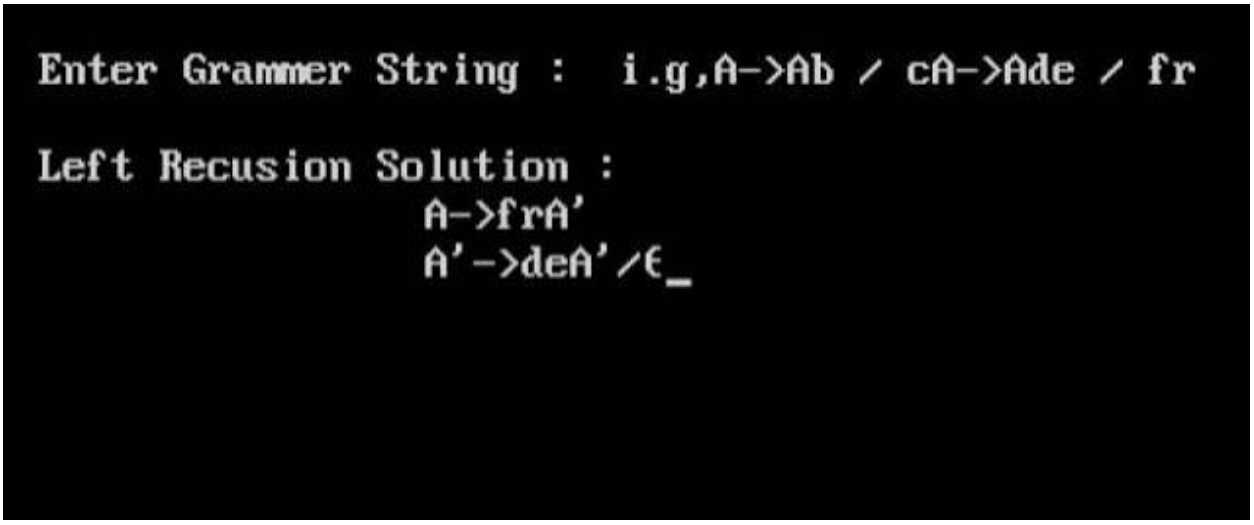
```
for(i=0;i<strlen(a1)||i<strlen(a2);i++)
{
if(a1[i]==a2[i]) {
a3[k]=a1[i];
k++;
}
else
{
a4[l]=a1[i];
a5[l]=a2[i];
l++;
}
}
a3[k]='X';
a3[++k]='\0';
a4[l]='/';
a5[l]='\0';
a4[++l]='\0';
strcat(a4,a5);
printf("\n A->%s",a3);
printf("\n X->%s",a4);
getch();
}
```


D:\5th sem\SP\Lex Programs\PR-9
enter any productions A->bcd/bcf
A->bcX
X->d/f

9. Write a program to remove the Left Recursion from a given grammar.

```
#include<stdio.h>

#include<conio.h>

void main()

{

        char l,r1[5],r2[5];

        int i,j=0;

        clrscr();

        printf("\nEnter Left Non-Terminal :\t");

        scanf("%c->%s / %s",&l,r1,r2);

        if(l==r1[0])

        {

         printf("\nLeft Recusion ");

         for(i=1;r1[i-1]!='\0';i++) r1[j++]=r1[i];

         printf("Solution :");

    printf("\n\t\t%c->%s%c'\n\t\t%c' >%s %c'/%c", l,r2,l,l,r1
,l,238);
```

```
        }
        if(l==r2[0]) {
                printf("\nLeft Recusion ");
                for(i=1;r2[i-1]!='\0';i++) r2[j++]=r2[i];
                printf("Solution :");
                printf("\n\t\t%c->%s %c'\n\t\t%c'-
>%s%c'/%c",l,r1,l,l,r2,l,238);
        }
        getch();
}
```



10. Implement Recursive Descendent Parsing for the given Grammar.

E -> T + E / T

T -> F * T / F

F -> (E) / i.

```
#include "stdio.h"

#include "conio.h"

char  input[100];

char prod[100][100];

int pos=-1,l,st=-1;

char id,num;

void E();

void T();

void F();

void advance();

void Td();

void Ed();

void advance()
```

```
{
pos++;
if(pos<l)
{
if(input[pos]>='0'&& input[pos]<='9')
{
num=input[pos];
id='\0';
}
if((input[pos]>='a' || input[pos]>='A')&&(input[pos]<='z' ||

input[pos]<='Z'))
{id=input[pos];
num='\0';
}
}
}
void E()
{
```

```
strcpy(prod[++st],"E->TE'");
T();
Ed();
}
void Ed()
{
int p=1;
if(input[pos]=='+')
{
p=0;
strcpy(prod[++st],"E'->+TE'");
advance();
T();
Ed();
}
if(input[pos]=='-')
{   p=0;
strcpy(prod[++st],"E'->-TE'");
```

```
advance();
T();
Ed();
}
```

// Recursive Descent

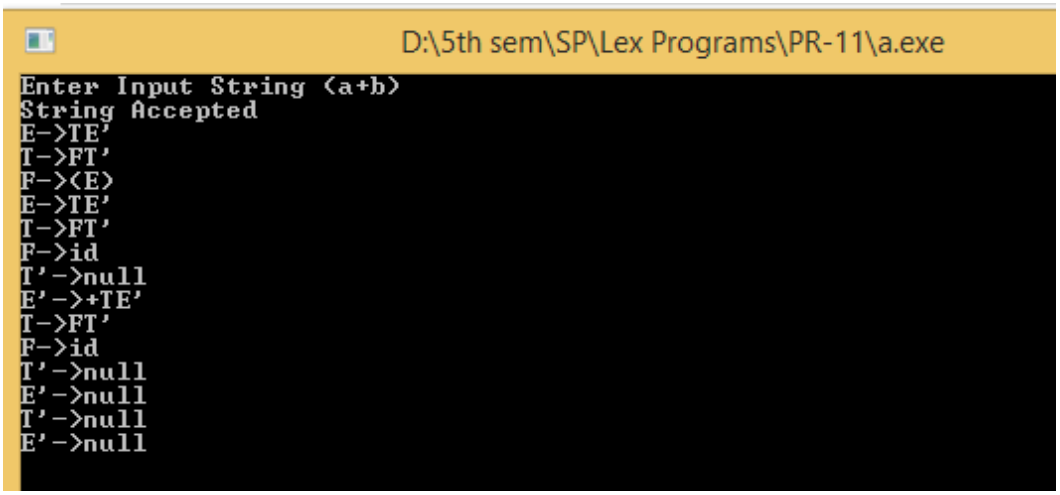
```
Parser
if(p==1)
{
strcpy(prod[++st],"E'->null");
}
}

void T()
{
strcpy(prod[++st],"T->FT'");
F();
Td();
```

```
}
void Td()
{
int p=1;
if(input[pos]=='*')
{
p=0;
strcpy(prod[++st],"T'->*FT'");
advance();
F();
Td();
}
if(input[pos]=='/')
{   p=0;
strcpy(prod[++st],"T'->/FT'");
advance();
F();
Td();
```

```
}
if(p==1)
strcpy(prod[++st],"T'->null");
}
void F()
{
if(input[pos]==id) {
strcpy(prod[++st],"F->id");
advance();        }
if(input[pos]=='(')
{
strcpy(prod[++st],"F->(E)");
advance();
E();
if(input[pos]==')')   {
//strcpy(prod[++st],"F->(E)");
advance();           }
}
```

```
if(input[pos]==num)
{
strcpy(prod[++st],"F->num");
advance();
}
}
int main()
{
int i;
printf("Enter Input String ");
scanf("%s",input);
l=strlen(input);
input[l]='$';
advance();
E();
if(pos==l)
{
printf("String Accepted\n");
```

```
for(i=0;i<=st;i++)
{
printf("%s\n",prod[i]);
}
}
else
{
printf("String rejected\n");
}
getch();
return 0;
}
```



11. Implement Predictive Parser for the given grammar.

E -> T + E / T

T -> F * T / F F -> (E) / i.

```
//To Implement Predictive Parsing
#include<string.h>
#include<conio.h>
char a[10];
int top=-1,i;
void error(){
printf("Syntax Error");
}
void push(char k[]) //Pushes The Set Of Characters on to
the Stack
{
  for(i=0;k[i]!='\0';i++)
  {
    if(top<9)
    a[++top]=k[i];
  }
}
char TOS()       //Returns TOP of the Stack
{
  return a[top];
```

```
}
void pop()      //Pops 1 element from the Stack
{
  if(top>=0)
    a[top--]='\0';
}
void display()  //Displays Elements Of Stack
{
  for(i=0;i<=top;i++)
    printf("%c",a[i]);
}
void display1(char p[],int m) //Displays The Present Input
String
{
  int l;
  printf("\t");
  for(l=m;p[l]!='\0';l++)
    printf("%c",p[l]);
}
char* stack(){
return a;
}
int main()
{
  char ip[20],r[20],st,an;
```

```
int ir,ic,j=0,k;
char t[5][6][10]={"$","$","TH","$","TH","$",
                "+TH","$","e","e","$","e",
                "$","$","FU","$","FU","$",
                "e","*FU","e","e","$","e",
                "$","$","(E)","$","i","$"};
clrscr();
printf("\nEnter any String(Append with $)");
gets(ip);
printf("Stack\tInput\tOutput\n\n");
push("$E");
display();
printf("\t%s\n",ip);
for(j=0;ip[j]!='\0';)
{
if(TOS()==an)
   {
      pop();
      display();
      display1(ip,j+1);
      printf("\tPOP\n");
      j++;
   }
  an=ip[j];
  st=TOS();
```

```
    if(st=='E')ir=0;
    else if(st=='H')ir=1;
    else  if(st=='T')ir=2;
    else if(st=='U')ir=3;
    else  if(st=='F')ir=4;
    else {
          error();
          break;
          }
    if(an=='+')ic=0;
    else if(an=='*')ic=1;
    else if(an=='(')ic=2;
    else if(an==')')ic=3;
    else
if((an>='a'&&an<='z')||(an>='A'&&an<='Z')){ic=4;an='i';}
    else if(an=='$')ic=5;
    strcpy(r,strrev(t[ir][ic]));
    strrev(t[ir][ic]);
    pop();
    push(r);
    if(TOS()=='e')
    {
      pop();
      display();
      display1(ip,j);
```

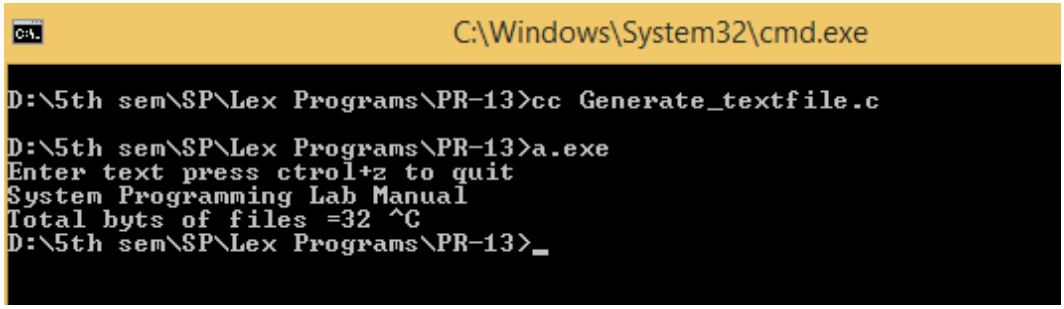
```
      printf("\t%c->%c\n",st,238);
    }
    else{
    display();
    display1(ip,j);
    printf("\t%c->%s\n",st,t[ir][ic]);
    }
    if(TOS()=='$'&&an=='$')
    break;
    if(TOS()=='$'){
      error();
      break;
      }
    }
    k=strcmp(stack(),"$");
    if(k==0 && i==strlen(ip))
  printf("\n Given String is accepted");
  else
  printf("\n Given String is not accepted");
 return 0;
}
```

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

D:\5th sem\SP\Lex Programs\PR-12>PredictiveParser.exe

Enter any String(Append with $)$A+B
Stack   Input   Output

$E      $A+B
$$      $A+B    E->$

 Given String is not accepted
D:\5th sem\SP\Lex Programs\PR-12>
```

12. Write a SAL program in text file and generate SYMTAB and LITTAB.

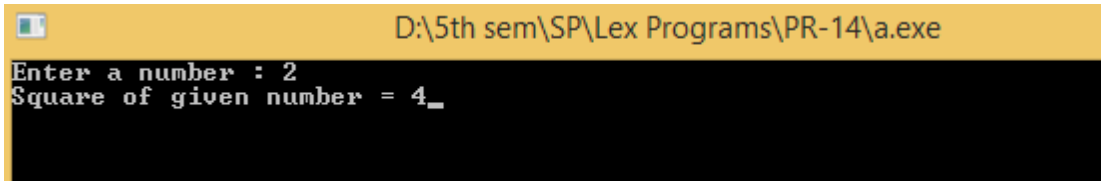
```
/* Create text file and read Text file and count Word and
Line
  display entire text on the screen */
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void main()
{
FILE *f;
char ch;
int line=0,word=0;
clrscr();
f=fopen("student","w+");
printf("Enter text press ctrol+z to quit\n");
do
{
ch=getchar();
putc(ch,f);
}
while(ch!=EOF);

printf("Total byts of files =%d ",ftell(f));
rewind(f);
```

```
while((ch=getc(f))!=EOF)
{
if(ch=='\n')
line++;
if(isspace(ch)||ch=='\t'||ch=='\n')
word++;
putchar(ch);
}
fclose(f);
printf("\n no of line=%d\n",line);
printf("no of word=%d\n",word);
getch();
}
```



13. Use macro features of C language.

```
#include<stdio.h>
#include<conio.h>
#define square(x) x*x
void main()
{
  int n,s;
  printf("Enter a number : ");
  scanf("%d",&n);
  s=square(n);
  printf("Square of given number = %d",s);
  getch();
}
```

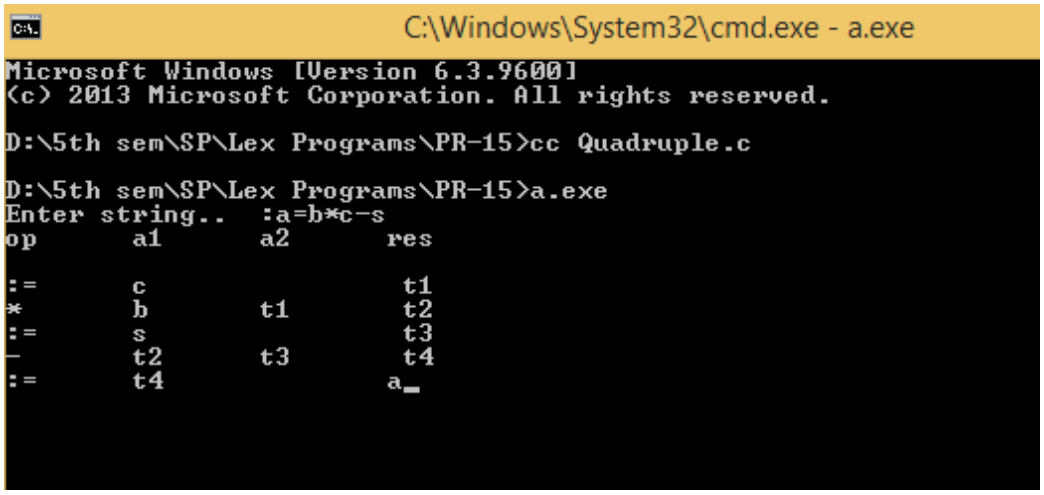


14. Write a program which generates Quadruple Table for the given postfix String.

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

main()
{

 char line[20];
int s[20];
int t=1;

int i=0;
printf("Enter string.. :");
gets(line);
for(i=0;i<20;i++)s[i]=0;
printf("op\ta1\ta2\tres\n");
for(i=2;line[i]!='\0';i++)
{
if(line[i]=='/' || line[i]=='*')
{
            printf("\n");
if(s[i]==0)
{
```

```
if(s[i+1]==0)
{
printf(":=\t%c\t\t t%d\n",line[i+1],t);
s[i+1]=t++;
}
printf("%c\t",line[i]);
(s[i-1]==0)?printf("%c\t",line[i-1]):printf("t%d\t",s[i-1]);
printf("t%d \t t%d",s[i+1],t);
s[i-1]=s[i+1]=t++;
s[i]=1;
}
}
}

for(i=2;line[i]!='\0';i++)
{
if(line[i]=='+' || line[i]=='-')
{
        printf("\n");
if(s[i]==0)
{
if(s[i+1]==0)
{
printf(":=\t%c\t\t t%d\n",line[i+1],t);
s[i+1]=t++;
```

```
}
printf("%c\t",line[i]);
(s[i-1]==0)?printf("%c\t",line[i-1]):printf("t%d\t",s[i-1]);
printf("t%d \t t%d",s[i+1],t);
s[i-1]=s[i+1]=t++;
s[i]=1;
}
}
}
printf("\n:=\tt%d\t\t%c",t-1,line[0]);

getch();

}
```



15. Write a C program to parse a given string using Predictive parsing for given grammar. type ? simple | ?id | array [simple] of type simple ? integer | char | num dotdot num.

```
//To Implement Predictive Parsing
#include<string.h>
#include<conio.h>
char a[10];
int top=-1,i;
void error(){
printf("Syntax Error");
}
void push(char k[]) //Pushes The Set Of Characters on to
the Stack
{
  for(i=0;k[i]!='\0';i++)
  {
    if(top<9)
    a[++top]=k[i];
  }
}
char TOS()       //Returns TOP of the Stack
{
```

```
  return a[top];
}
void pop()      //Pops 1 element from the Stack
{
  if(top>=0)
    a[top--]='\0';
}
void display()  //Displays Elements Of Stack
{
  for(i=0;i<=top;i++)
    printf("%c",a[i]);
}
void display1(char p[],int m) //Displays The Present Input
String
{
  int l;
  printf("\t");
  for(l=m;p[l]!='\0';l++)
    printf("%c",p[l]);
}
char* stack(){
return a;
}
int main()
{
```

```
char ip[20],r[20],st,an;
int ir,ic,j=0,k;
char t[5][6][10]={"$","$","TH","$","TH","$",
                  "+TH","$","e","e","$","e",
                  "$","$","FU","$","FU","$",
                  "e","*FU","e","e","$","e",
                  "$","$","(E)","$","i","$"};
clrscr();
printf("\nEnter any String(Append with $)");
gets(ip);
printf("Stack\tInput\tOutput\n\n");
push("$E");
display();
printf("\t%s\n",ip);
for(j=0;ip[j]!='\0';)
{
if(TOS()==an)
   {
     pop();
     display();
     display1(ip,j+1);
     printf("\tPOP\n");
     j++;
   }
  an=ip[j];
```

```
st=TOS();
 if(st=='E')ir=0;
 else if(st=='H')ir=1;
 else  if(st=='T')ir=2;
 else if(st=='U')ir=3;
 else  if(st=='F')ir=4;
 else {
       error();
       break;
       }
 if(an=='+')ic=0;
 else if(an=='*')ic=1;
 else if(an=='(')ic=2;
 else if(an==')')ic=3;
 else
if((an>='a'&&an<='z')||(an>='A'&&an<='Z')){ic=4;an='i';}
 else if(an=='$')ic=5;
 strcpy(r,strrev(t[ir][ic]));
 strrev(t[ir][ic]);
 pop();
 push(r);
 if(TOS()=='e')
 {
   pop();
   display();
```

```
      display1(ip,j);
      printf("\t%c->%c\n",st,238);
    }
    else{
    display();
    display1(ip,j);
    printf("\t%c->%s\n",st,t[ir][ic]);
    }
    if(TOS()=='$'&&an=='$')
    break;
    if(TOS()=='$'){
      error();
      break;
      }
    }
    k=strcmp(stack(),"$");
    if(k==0 && i==strlen(ip))
  printf("\n Given String is accepted");
  else
  printf("\n Given String is not accepted");
 return 0;
}
```

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

D:\5th sem\SP\Lex Programs\PR-12>PredictiveParser.exe

Enter any String(Append with $)$A+B
Stack   Input   Output

$E      $A+B
$$      $A+B    E->$

 Given String is not accepted
D:\5th sem\SP\Lex Programs\PR-12>
```