

## Programs

### 1a. Program to count the number, words, spaces and lines in a given input file

Code:

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

int c=0;
int w=0;
int s=0;
int l=0;
}%
%%
" " { s++; w++;}
[\n] { l++; w++;}
[\t\n] {w++;}
[^\t\n] {c++;}
%%
int yywrap()
{
return 1;
}
int main()
{
yyin=fopen("Info.txt", "r");
yylex();
printf("Characters = %d\nWords = %d\nSpaces = %d\nLines= %d\n",c,w,s,l);
return 0;
}
```

Input file:

```
Programs/Info.txt
1 Hello
2 This is an information text
```

Output:

```
Info.txt
Terminal
~/Programs$ lex Prog1a.l
~/Programs$ cc lex.yy.c -ll
~/Programs$ ./a.out
Characters = 28
Words = 6
Spaces = 4
Lines= 2
~/Programs$
```

### 1b. Program to recognize and count the number of identifiers in a file

Code:

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

int count=0,ch=0,wd=0,sp=0,ln=0;
%}

digit [0-9]
letter [a-zA-Z_]

%%

{letter}({letter}|{digit})* {count++;}

.+ {}

\n {}

%%

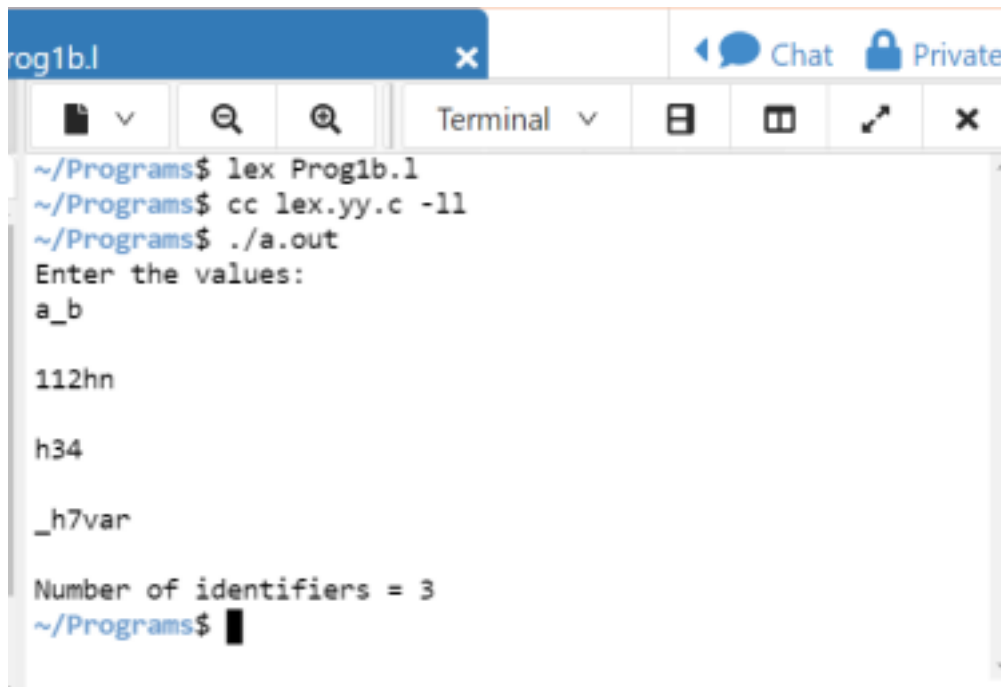
int main()
```

```

{
yyin=fopen("1b.txt","r");
yylex();
printf("count=%d",count);
return 0;
}
int yywrap()
{
return 1;
}

```

Output:



```

Prog1b.l x
Chat Private
Terminal
~/Programs$ lex Prog1b.l
~/Programs$ cc lex.yy.c -ll
~/Programs$ ./a.out
Enter the values:
a_b
112hn
h34
_h7var
Number of identifiers = 3
~/Programs$

```

**2a. Programs to count the numbers of comments lines in a given C program. Also eliminate them and copy the resulting program into separate file.**  
**Code**

```

%{
#include<stdio.h>

```

```

int s=0,m=0;

%}

%%

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

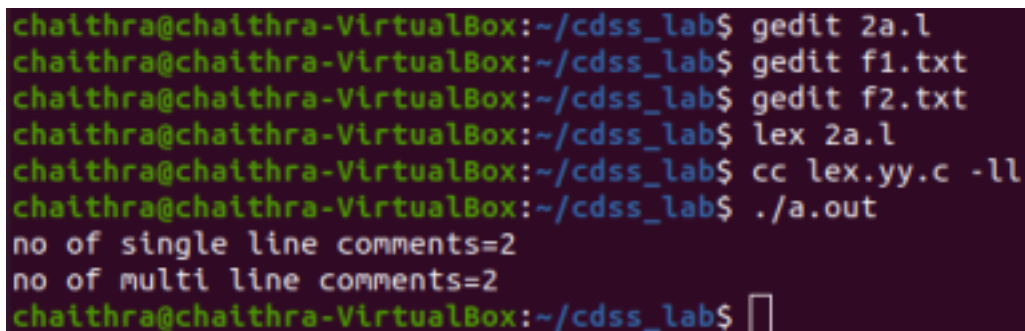
/*".* s++;

%%

void main(){
yyin=fopen("f1.txt","r");
yyout=fopen("f2.txt","w");
yylex();
fclose(yyin);
fclose(yyout);
printf("no of single line comments=%d\n",s);
printf("no of multi line comments=%d\n",m);
}

int yywrap()
{
return 1;
}

```



```

chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit 2a.l
chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit f1.txt
chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit f2.txt
chaithra@chaithra-VirtualBox:~/cdss_lab$ lex 2a.l
chaithra@chaithra-VirtualBox:~/cdss_lab$ cc lex.yy.c -ll
chaithra@chaithra-VirtualBox:~/cdss_lab$ ./a.out
no of single line comments=2
no of multi line comments=2
chaithra@chaithra-VirtualBox:~/cdss_lab$ 

```

Input file f1.txt with comments and C program:

```
Open  ▾  [?]  f1.txt
~/cdss_lab

1 /* write a c
2 program
3 to print
4 hello world*/
5 void main();//main function
6 {
7 printf("Hello World!");//printing hello world
8 }
9 /* This will be the
10 output of program*/
```

Output file with only C program:

```
Open  ▾  [?]  f2.txt
~/cdss_lab

1
2 void main()
3 {
4 printf("Hello World!");
5 }
6
```

## 2b. Program to recognize whether a given sentence is simple or compound.

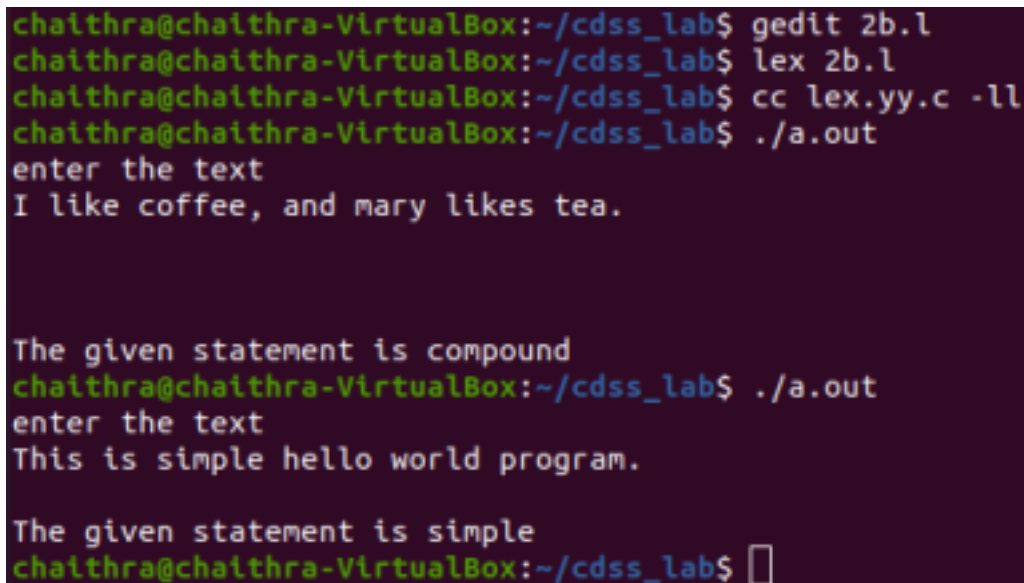
Code:

```
%{
#include<stdio.h>
int c=0;
%}
%%
[a-zA-Z]*[ ](and|or|but|yet|so)[ ] [a-zA-Z]* {c=1;}
.|\n];
%%
int yywrap()
{
```

```

return 1;
}
void main(){
printf("enter the text\n");
yylex();
if(c)
{
printf("The given statement is compound\n");
}
else
{
printf("The given statement is simple\n");
}
}

```



```

chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit 2b.l
chaithra@chaithra-VirtualBox:~/cdss_lab$ lex 2b.l
chaithra@chaithra-VirtualBox:~/cdss_lab$ cc lex.yy.c -ll
chaithra@chaithra-VirtualBox:~/cdss_lab$ ./a.out
enter the text
I like coffee, and mary likes tea.

The given statement is compound
chaithra@chaithra-VirtualBox:~/cdss_lab$ ./a.out
enter the text
This is simple hello world program.

The given statement is simple
chaithra@chaithra-VirtualBox:~/cdss_lab$ 

```

**3a. Program to count number 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("Number of positive integer %d\n",pi);
printf("Number of negative integer %d\n",ni);
printf("Number of positive fraction %d\n",pf);
printf("Number of negative fraction %d\n",nf);
}

int yywrap(){
return 1;
}

```

```
chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit 3a.l
chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit in.txt
chaithra@chaithra-VirtualBox:~/cdss_lab$ lex 3a.l
chaithra@chaithra-VirtualBox:~/cdss_lab$ cc lex.yy.c -ll
chaithra@chaithra-VirtualBox:~/cdss_lab$ ./a.out in.txt
```

```
Number of positive integer 2
Number of negative integer 2
Number of positive fraction 2
Number of negative fraction 1
chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit 3a.l
```

Input file with values:

A screenshot of a text editor window titled 'in.txt' with the path '-/cdss\_lab'. The editor contains seven lines of text, each representing a number with its classification: '1 1', '2 2', '3 -64', '4 -8', '5 8.73', '6 3.92', and '7 -8.38'. The seventh line is highlighted with a light gray background.

```
1 1
2 2
3 -64
4 -8
5 8.73
6 3.92
7 -8.38
```

**3b. Program to count the number of “scanf” and “printf” statements in a C program. Replace them with “readf” and “writef” statements respectively.**

Code:

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

int sf=0,pf=0;

%}

%%

"scanf" {sf++; fprintf(yyout,"readf");}
```



```

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

int main()
{
yyin=fopen("f1.c","r");
yyout=fopen("f2.c","w");
yylex();
printf("no of scanf =%d\n no of printf =%d\n",sf,pf);
return 0;
}

```

```

chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit 3b.l
chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit f1.c
chaithra@chaithra-VirtualBox:~/cdss_lab$ gedit f2.c
chaithra@chaithra-VirtualBox:~/cdss_lab$ lex 3b.l
chaithra@chaithra-VirtualBox:~/cdss_lab$ cc lex.yy.c -ll
chaithra@chaithra-VirtualBox:~/cdss_lab$ ./a.out
no of scanf =1
no of printf =2
chaithra@chaithra-VirtualBox:~/cdss_lab$ 

```

Input file with printf and scanf statements:



```

Open ▼ [+/-] f1.c ~/cdss_lab
1 #include<stdio.h>
2 int main()
3 {
4 int a,b,c;
5 printf("enter the values of a and b\n");
6 scanf("%d%d",&a,&b);
7 c=a+b;
8 printf("Sum=%d",c);
9 return 0;
10

```

Output file with readf and writef statements:

```
Open  ▾  [icon]  f2.c  ~/cdss_lab

1 #include<stdio.h>
2 int main()
3 {
4     int a,b,c;
5     writef("enter the values of a and b\n");
6     readf("%d%d",&a,&b);
7     c=a+b;
8     writef("Sum=%d",c);
9     return 0;
10 }
```

f1.c

```
/* This is the simple c program
to add two numbers */
```

```
#include <stdio.h>
int main()
//printing two numbers
{
    int num1, num2, sum;
    printf("Enter first number: ");
    //prints first number
    scanf("%d", &num1);
    printf("Enter second number: ");
    scanf("%d", &num2);

    sum = num1 + num2;
    printf("Sum of the entered numbers: %d", sum);
    return 0;
}
```

Type gedit f2.c u can find output over there in read and write f manner in place of printf and scanf manner

#### 4. Program to evaluate arithmetic expression involving operators +, -, \*, / Code:

Code:

Lex:

```
//lex code
%{
#include "y.tab.h"

extern yyval;

%}

%%

[0-9]+ {yyval=atoi(yytext);return num;}

[+\-\\*\V] {return yytext[0];}

[] {return yytext[0];}

[(] {return yytext[0];}

. {;}

\n {return 0;}

%%

//yacc code
%{#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 {$$=$1/$3;}
```

```

|exp/'exp {if($3==0){printf("Division by zero\n");exit(0);}
        else
            $$=$1/$3;}
|('exp') {$$=$2;}
|num {$$=$1;};
%%

```

```

int yyerror()
{
printf("error");
exit(0);
}

int main()
{
printf("Enter the expression:\n");
yyparse();
}

```

Output:

```

~/Programs$ ./a.out
Enter the expression:
6*7
42
~/Programs$ ./a.out
Enter the expression:
5/0
Division by zero
~/Programs$ █

```

**5. Program to recognize a valid variable which starts with a letter, followed by any number of letter or digits**

Code:

```
//lex code

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

%%

[a-zA-z] return L;
[0-9] return D;
%%
int yywrap(void) {return 1;}
```

```
//yacc code

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

%token L D
```

```
%%

var:L E {printf("Valid Variable\n"); return 0;}
E:E L;
|E D;
| ;
%%
```

```
int main()
{
printf("Type the variable\n");
yyparse();
return 0;
```

```

}
int yyerror()
{
printf("Invalid Variable\n");
exit(0);
}

```

Output:

```

~/Programs$ ./a.out
Type the variable
vansd123

Valid Variable
~/Programs$ ./a.out
Type the variable
90vans
Invalid Variable
~/Programs$ █

```

## 6. Program to recognize the strings using the grammar ( $a^n b^n$ ; $n \geq 0$ )

Code:

```

//lex code
%{
#include "y.tab.h"
%}
%%

a return A;
b return B;
. return yytext[0];
\n return yytext[0];
%%

int yywrap(void) {return 1;}

```

```
//yacc code
```

```
%{
```

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

```
#include<stdlib.h>
```

```
%}
```

```
%token A B
```

```
%%
```

```
Str:S '\n' {return 0;}
```

```
S:A S B;
```

```
|;
```

```
%%
```

```
int main()
```

```
{
```

```
printf("Type the string\n");
```

```
if (!yyparse())
```

```
printf("Valid String\n");
```

```
return 0;
```

```
}
```

```
int yyerror()
```

```
{
```

```
printf("Invalid String\n");
```

```
exit(0);
```

```
}
```

Output:

```

~/Programs$ ./a.out
Type the string
abb
Invalid String
~/Programs$ ./a.out
Type the string
aabb
Valid String
~/Programs$ ./a.out
Type the string
abab
Invalid String
~/Programs$ ./a.out
Type the string
a b
Invalid String
~/Programs$ █

```

## 7. C program to implement Pass1 of Assembler Code:

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

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

```
#include<stdlib.h>
```

```
void main()
```

```
{
```

```
    char opcode[10], operand[10], label[10], code[10], mnemonic[3]; int
    locctr, start, length;
```

```
    FILE *fp1,*fp2,*fp3,*fp4;
```

```
    fp1=fopen("Input.txt","r");
```

```
    fp2=fopen("Optab.txt","r");
```

```
    fp3=fopen("Symtabl.txt","w");
```

```
    fp4=fopen("Out.txt","w");
```



```
fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
```

```
if(strcmp(opcode, "START")==0)
```

```
{
```

```
start=atoi(operand);
```

```
locctr=start;
```

```
fprintf(fp4, "\t%s\t%s\t%s\n", label, opcode, operand);
```

```
fscanf(fp1, "%s\t%s\t%s", label, opcode, operand); }
```

```
else
```

```
locctr=0;
```

```
while(strcmp(opcode, "END")!=0)
```

```
{
```

```
fprintf(fp4, "%d\t", locctr);
```

```
if(strcmp(label, "**")!=0)
```

```
fprintf(fp3, "%s\t%d\n", label, locctr);
```

```
fscanf(fp2, "%s\t%s", code, mnemonic);
```

```
while(strcmp(code, "END")!=0)
```

```
{
```

```
if(strcmp(opcode, code)==0)
```

```
{
```

```
locctr+=3;
```

```
break;
```

```
}
```

```
fscanf(fp2, "%s\t%s", code, mnemonic); }
```

```
if(strcmp(opcode, "WORD")==0)
```

```
locctr+=3;
```

```
else if(strcmp(opcode, "RESW")==0)
```

```

locctr+=(3*(atoi(operand)));
else if(strcmp(opcode,"RESB")==0)
locctr+=atoi(operand);
else if(strcmp(opcode,"BYTE")==0)
++locctr;
fprintf(fp4, "%s\t%s\t%s\t\n",label,opcode,operand);
fscanf(fp1, "%s\t%s\t%s",label,opcode,operand); }

fprintf(fp4, "%d\t%s\t%s\t%s\n",locctr,label,opcode,operand);
length=locctr-start;

printf("The length of the code:%d\n",length);


fclose(fp1);

fclose(fp2);
fclose(fp3);

fclose(fp4);

}

```

Input files:

Input files:

1. Input.txt

Programs/Input.txt			
1	**	START	2000
2	**	LDA	FIVE
3	**	STA	ALPHA
4	**	LDCH	CHARZ
5	**	STCH	C1
6	ALPHA	RESW	2
7	FIVE	WORD	5
8	CHARZ	BYTE	C'Z'
9	C1	RESB	1
10	**	END	**

2. Optab.txt

Programs/Optab.txt		
1	START	*
2	LDA	03
3	STA	0f
4	LDCH	53
5	STCH	57
6	END	*

Output:

```
~/Programs$ ./a.out
```

```
The length of the code:23
```

```
~/Programs$ █
```



Output files:

1.Symtabl.txt

Programs/Symtabl.txt		
1	ALPHA	2012
2	FIVE	2018
3	CHARZ	2021
4	C1	2022
5		

3. Out.txt

Programs/Out.txt				
1		**	START	2000
2	2000	**	LDA	FIVE
3	2003	**	STA	ALPHA
4	2006	**	LDCH	CHARZ
5	2009	**	STCH	C1
6	2012	ALPHA	RESW	2
7	2018	FIVE	WORD	5
8	2021	CHARZ	BYTE	C'Z'
9	2022	C1	RESB	1
10	2023	**	END	**
11				

### **Input.txt**

```
**  START  2000  
  
**  LDA   FIVE  
  
**  STA   ALPHA  
  
**  LDCH  CHARZ  
  
**  STCH  C1  
  
ALPHA RESW  2  
  
FIVE  WORD  5  
  
CHARZ BYTE  C'Z'  
  
C1    RESB  1  
  
**  END   **
```

### **Optab.txt**

```
START *  
  
LDA  03  
  
STA  0f  
  
LDCH 53
```

**STCH 57**

**END \***

**Symtabl.txt**

**ALPHA 2012**

**FIVE 2018**

**CHARZ 2021**

**C1 2022**

**Out.txt**

```

    **      START      2000

2000 **      LDA  FIVE

2003 **      STA  ALPHA

2006 **      LDCH CHARZ

2009 **      STCH C1

2012 ALPHA      RESW2

2018 FIVE WORD      5

2021 CHARZ      BYTE C'Z'

2022 C1      RESB 1
```

**2023    \*\*        END    \*\***

### **8. C program to implement Absolute loader.**

Code:

```
#include<stdio.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];
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[i]='\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';
        staddrl=atoi(staddr);
        i=12;
        while(line[i]!='$')
        {
            if(line[i]!='^')
            {
                printf("00%d\t%c%c\n",staddrl,line[i],line[i+1]);
                staddrl++;
                i=i+2;
            }
            else
                i++;
        }
        else if(line[0]=='E')
            fclose(fp);
    }
    while(!feof(fp));
}
}

```

**Input file:**

**H^SAMPLE^001000^0035**

**T^001000^0C^001003^071009\$**

**T^002000^03^111111\$**

**E^001000**

```
}  
}
```

```
student@lab-OptiPlex-3020:~$ gedit a.c  
student@lab-OptiPlex-3020:~$ gcc a.c  
student@lab-OptiPlex-3020:~$ ./a.out  
enter program anmeSAMPLE  
name from obj.SAMPLE  
001000 00  
001001 10  
001002 03  
001003 07  
001004 10  
001005 09  
002000 11  
002001 11  
002002 11
```

## 9. C program to implement the FIRST in context free grammar

Code:

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

```
void FIRST(char);
```

```
int count,n=0;
```

```
char prodn[10][10],first[10];
```

```
void main()
```

```
{
```

```
int i,choice;
```

```
char c,ch;
```

```
printf("Enter the number of productions: ");
```

```
scanf("%d",&count);
```

```

printf("Enter %d productions:\nEpsilon=$\n",count);
for(i=0;i<count;i++)

scanf("%s%c",prodn[i],&ch);

do{
n=0;

printf("Element :");

scanf("%c",&c);

FIRST(c);

printf("\nFIRST(%c)={",c);

for(i=0;i<n;i++)

printf(" %c",first[i]);

printf("}\n");

printf("Press 1 to continue :");

scanf("%d%c",&choice,&ch);

}

while(choice==1);

}

```

```

void FIRST(char c)
{
int j;
if(!(isupper(c)))first[n++]=c;
for(j=0;j<count;j++)
{
if(prodn[j][0]==c)
{
if(prodn[j][2]=='S')first[n++]='$';
else if(islower(prodn[j][2]))first[n++]=prodn[j][2];
else FIRST(prodn[j][2]);

```

}  
}  
}

## OUTPUT:

```
~/Programs$ ./a.out
Enter the number of productions: 8
Enter 8 productions:
Epsilon=$
E=TD
D=+TD
D=$
T=FS
S=*FS
S=$
F=(E)
F=a
Element :E

FIRST(E)={ ( a}
Press 1 to continue :1
Element :D

FIRST(D)={ + $}
Press 1 to continue :1
Element :T

FIRST(T)={ ( a}
Press 1 to continue :1
Element :S

FIRST(S)={ * $}
Press 1 to continue :0
~/Programs$ █
```

## 10. C program to implement Shift Reduce Parser for the given grammar: E

→ E+E

E → E\*E

E → (E)

E → id

Code:

```
#include<stdio.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();

int main()
{ 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");
printf("\n$\t%s$\t---",a);
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]=a[j+1];
stk[i+2]='\0';
a[j]=' ';
printf("\n%s$\t%s$\t%ssymbols",stk,a,act);
check(); }
}
}

```

```

}
void check()
{
strcpy(ac,"REDUCE TO E");
for(z=0; z<c; z++)
if(stk[z]=='i' && stk[z+1]=='d')
{ stk[z]='E';
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;
} }

```

Output:

```

C:\Users\HAASINI\OneDrive\Documents\6thsem\lab\lab10.exe
GRAMMAR is E->E+E
E->E*E
E->(E)
E->id
Enter input string
id+id*id
stack  input  action
$      id+id*id$  ---
$id    +id*id$    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  $           SHIFT->id
$E*E   $           REDUCE TO E
$E     $           REDUCE TO E
Process returned 0 (0x0)   execution time : 30.724 s
Press any key to continue.

```