LR(0)

NAME : ATHRESH KUMAR LABDE
RA1911033010146
M1

AIM : To verify leading and trailing in this experiment by implementing and running the code.

ALGORITHM :
Input − Context Free Grammar G

Output − LEADING (A) = {a} iff Boolean Array L [A, a] = true

Method − Procedure Install (A, a) will make L (A, a) to true if it was not true earlier.

```
begin

For each non-terminal A and terminal a

                L [A, a] = false ;

For each production of form A ⟶ aα or A → B a α
                Install (A, a) ;

While the stack not empty
                Pop top pair (B, a) form Stack ;

                 For each production of form A → B α

                 Install (A, a);

end
```

Procedure Install (A, a)

```
begin
If not L [A, a] then
            L [A, a] = true

            push (A, a) onto stack.

end
```

Algorithm to compute TRAILING

Input − Context Free Grammar G

Output − TRAILING (A) = {a} iff Boolean Array T [A, a] = true

Method

```
begin
For each non-terminal A and terminal a
        T [A, a] = false ;

For each production of form A → αa or A → α a B
        Install (A, a) ;

While the stack not empty
        Pop top pair (B, a) form Stack ;

        For each production of form A → αB

        Install (A, a);

end
Procedure Install (A, a)

begin
If not T [A, a] then
        T [A, a] = true

        push (A, a) onto stack.

end
```

Algorithm for Computing Operator Precedence Relations

Input − An Operator Grammar

Output − A Precedence Relations between terminals and symbols.

Method

```
begin
For each production A → B1, B2, … … … . Bn
            for i = 1 to n – 1

       If Bi and Bi+1 are both terminals then

            set Bi = Bi+1

       If i ≤ n − 2 and Bi and Bi+2are both terminals and Bi+1 is non-terminal then

            set Bi = Bi+1

       If Biis terminal & Bi+1is non-terminal then for all a in LEADING (Bi+1)
```

```
            set Bi <. a

        If Biis non-terminal & Bi+1 is terminal then for all a in TRAILING (Bi)

            set a . > Bi+1
End
```

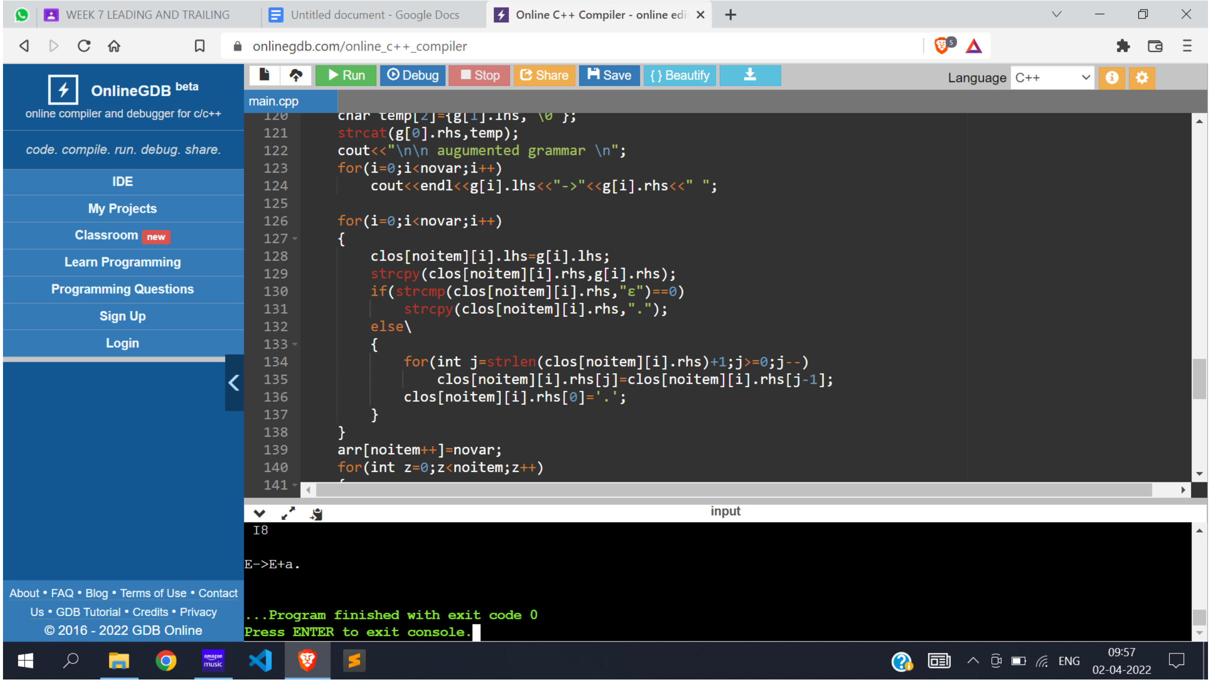
CODE :

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

using namespace std;

char prod[20][20],listofvar[26]="ABCDEFGHIJKLMNOPQR";
int novar=1,i=0,j=0,k=0,n=0,m=0,arr[30];
int noitem=0;

struct Grammar
{
	char lhs;
	char rhs[8];
}g[20],item[20],clos[20][10];

int isvariable(char variable)
{
	for(int i=0;i<novar;i++)
		if(g[i].lhs==variable)
			return i+1;
	return 0;
}
void findclosure(int z, char a)
{
	int n=0,i=0,j=0,k=0,l=0;
	for(i=0;i<arr[z];i++)
	{
		for(j=0;j<strlen(clos[z][i].rhs);j++)
		{
			if(clos[z][i].rhs[j]=='.' && clos[z][i].rhs[j+1]==a)
			{
				clos[noitem][n].lhs=clos[z][i].lhs;
				strcpy(clos[noitem][n].rhs,clos[z][i].rhs);
				char temp=clos[noitem][n].rhs[j];
				clos[noitem][n].rhs[j]=clos[noitem][n].rhs[j+1];
				clos[noitem][n].rhs[j+1]=temp;
				n=n+1;
			}
		}
```

```
    }
    for(i=0;i<n;i++)
    {
        for(j=0;j<strlen(clos[noitem][i].rhs);j++)
        {
            if(clos[noitem][i].rhs[j]=='.' && isvariable(clos[noitem][i].rhs[j+1])>0)
            {
                for(k=0;k<novar;k++)
                {
                    if(clos[noitem][i].rhs[j+1]==clos[0][k].lhs)
                    {
                        for(l=0;l<n;l++)
                            if(clos[noitem][l].lhs==clos[0][k].lhs &&
strcmp(clos[noitem][l].rhs,clos[0][k].rhs)==0)
                                break;
                        if(l==n)
                        {
                            clos[noitem][n].lhs=clos[0][k].lhs;
                        strcpy(clos[noitem][n].rhs,clos[0][k].rhs);
                            n=n+1;
                        }
                    }
                }
            }
        }
    }
    arr[noitem]=n;
    int flag=0;
    for(i=0;i<noitem;i++)
    {
        if(arr[i]==n)
        {
            for(j=0;j<arr[i];j++)
            {
                int c=0;
                for(k=0;k<arr[i];k++)
                    if(clos[noitem][k].lhs==clos[i][k].lhs &&
strcmp(clos[noitem][k].rhs,clos[i][k].rhs)==0)
                        c=c+1;
                if(c==arr[i])
                {
                    flag=1;
                    goto exit;
                }
            }
        }
    }
    exit:;
    if(flag==0)
        arr[noitem++]=n;
}

int main()
{
    cout<<"ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :\n";
    do
    {
        cin>>prod[i++];
    }while(strcmp(prod[i-1],"0")!=0);
    for(n=0;n<i-1;n++)
    {
        m=0;
        j=novar;
        g[novar++].lhs=prod[n][0];
        for(k=3;k<strlen(prod[n]);k++)
        {
            if(prod[n][k] != '|')
            g[j].rhs[m++]=prod[n][k];
            if(prod[n][k]=='|')
            {
                g[j].rhs[m]='\0';
                m=0;
                j=novar;
                g[novar++].lhs=prod[n][0];
            }
        }
    }
    for(i=0;i<26;i++)
        if(!isvariable(listofvar[i]))
            break;
    g[0].lhs=listofvar[i];
    char temp[2]={g[1].lhs,'\0'};
    strcat(g[0].rhs,temp);
    cout<<"\n\n augumented grammar \n";
    for(i=0;i<novar;i++)
        cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";

    for(i=0;i<novar;i++)
    {
        clos[noitem][i].lhs=g[i].lhs;
        strcpy(clos[noitem][i].rhs,g[i].rhs);
        if(strcmp(clos[noitem][i].rhs,"ε")==0)
            strcpy(clos[noitem][i].rhs,".");
        else\
        {
            for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
```

```
                clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j-1];
            clos[noitem][i].rhs[0]='.';
        }
    }
    arr[noitem++]=novar;
    for(int z=0;z<noitem;z++)
    {
        char list[10];
        int l=0;
        for(j=0;j<arr[z];j++)
        {
            for(k=0;k<strlen(clos[z][j].rhs)-1;k++)
            {
                if(clos[z][j].rhs[k]=='.')
                {
                    for(m=0;m<l;m++)
                        if(list[m]==clos[z][j].rhs[k+1])
                            break;
                    if(m==l)
                        list[l++]=clos[z][j].rhs[k+1];
                }
            }
        }
        for(int x=0;x<l;x++)
            findclosure(z,list[x]);
    }
    cout<<"\n THE SET OF ITEMS ARE \n\n";
    for(int z=0; z<noitem; :z++)
    {
        cout<<"\n I"<<z<<"\n\n";

        for(j=0;j<arr[z];j++)
            cout<<clos[z][j].lhs<<"->"<<clos[z][j].rhs<<"\n";
        if(z==1){
          cout<<"Special output\n";
          cout<<clos[1][0].lhs<<"->"<<clos[1][0].rhs<<"\n";
          cout<<clos[5][0].lhs<<"->"<<clos[5][0].rhs<<"\n";
        }

    }
}
```

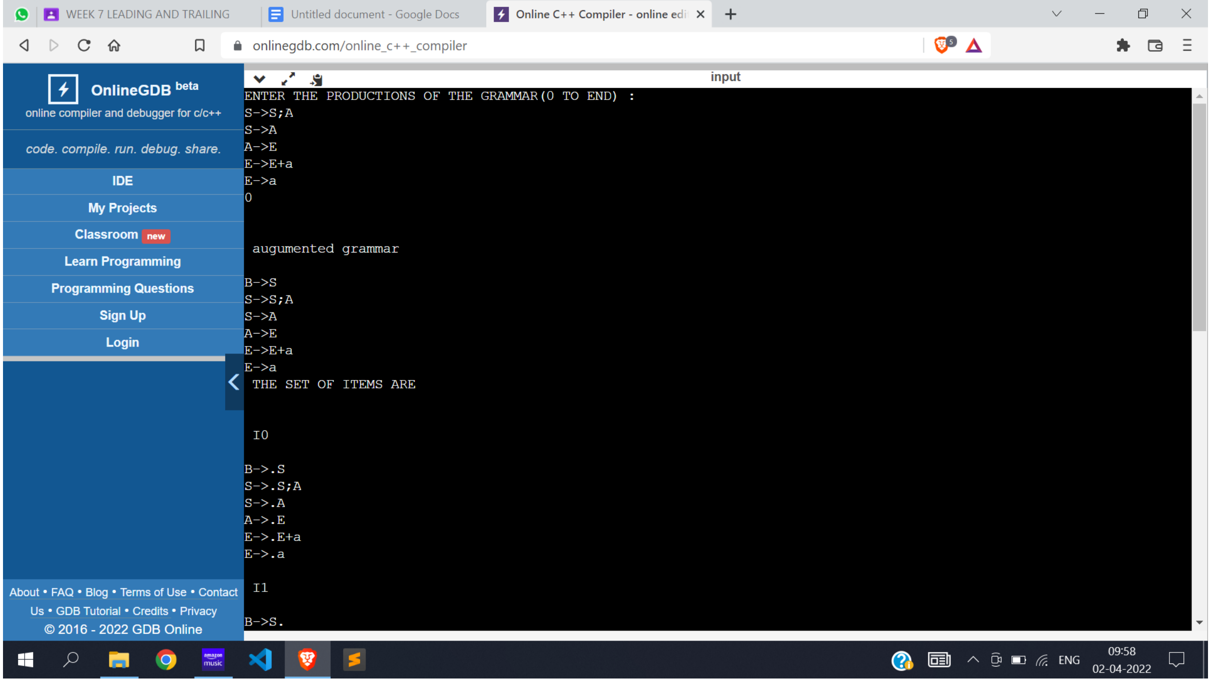
CODE SCREENSHOT :



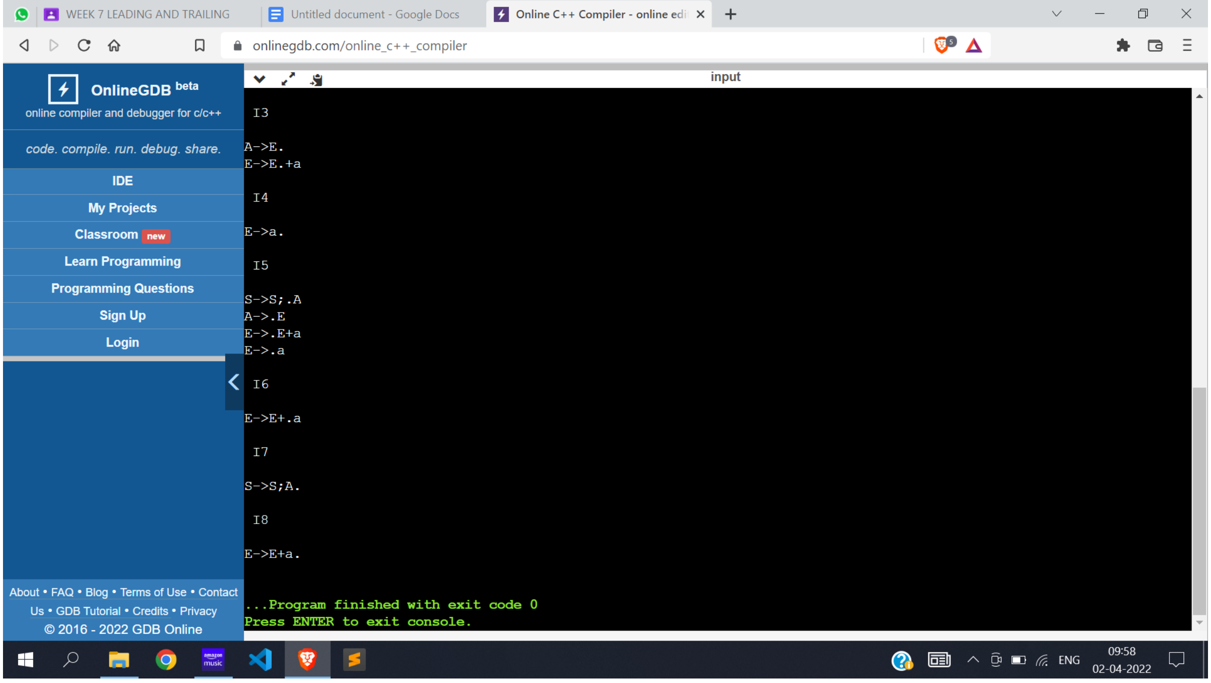
```
    char temp[2]={g[1].lhs,'\0'};
    strcat(g[0].rhs,temp);
    cout<<"\n\n augumented grammar \n";
    for(i=0;i<novar;i++)
        cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";

    for(i=0;i<novar;i++)
    {
        clos[noitem][i].lhs=g[i].lhs;
        strcpy(clos[noitem][i].rhs,g[i].rhs);
        if(strcmp(clos[noitem][i].rhs,"ε")==0)
            strcpy(clos[noitem][i].rhs,".");
        else\
        {
            for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
                clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j-1];
            clos[noitem][i].rhs[0]='.';
        }
    }
    arr[noitem++]=novar;
    for(int z=0;z<noitem;z++)
```

```
 I8

E->E+a.


...Program finished with exit code 0
Press ENTER to exit console.
```

OUTPUT :



```
ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :
S->S;A
S->A
A->E
E->E+a
E->a
0


 augumented grammar

B->S
S->S;A
S->A
A->E
E->E+a
E->a
 THE SET OF ITEMS ARE


 I0

B->.S
S->.S;A
S->.A
A->.E
E->.E+a
E->.a

 I1

B->S.
```



WEEK 7 LEADING AND TRAILING
Untitled document - Google Docs
Online C++ Compiler - online edi
onlinegdb.com/online_c++_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
input
ENG
09:58
02-04-2022

```
 I3

A->E.
E->E.+a

 I4

E->a.

 I5

S->S;.A
A->.E
E->.E+a
E->.a

 I6

E->E+.a

 I7

S->S;A.

 I8

E->E+a.


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT : hence we have successfully verified the LR(0)experiment by implementing and running the code.