**CD EXP 9 – LR & SLR**

**Aman Kalla**

**RA1911003010640**

**LR 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";

}

}

**SLR CODE:**

#include<stdio.h>

#include<string.h>

int axn[][6][2]={

{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

{{-1,-1},{100,6},{-1,-1},{-1,-1},{-1,-1},{102,102}},

{{-1,-1},{101,2},{100,7},{-1,-1},{101,2},{101,2}},

{{-1,-1},{101,4},{101,4},{-1,-1},{101,4},{101,4}},

{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

{{-1,-1},{101,6},{101,6},{-1,-1},{101,6},{101,6}},

{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

{{-1,-1},{100,6},{-1,-1},{-1,-1},{100,1},{-1,-1}},

{{-1,-1},{101,1},{100,7},{-1,-1},{101,1},{101,1}},

{{-1,-1},{101,3},{101,3},{-1,-1},{101,3},{101,3}},

{{-1,-1},{101,5},{101,5},{-1,-1},{101,5},{101,5}}

};//Axn Table

int

gotot[12][3]={1,2,3,-1,-1,-1,-1,-1,-1,-1,-1,-1,8,2,3,-1,-1,-1,-1,9,3,-1,-1,10,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};

//GoTo table

int a[10];

char b[10];

int top=-1,btop=-1,i;

void push(int k)

{

if(top<9)

a[++top]=k;

}

void pushb(char k)

{

if(btop<9)

b[++btop]=k;

}

char TOS()

{

return a[top];

}

void pop()

{

if(top>=0)

top--;

}

void popb()

{

if(btop>=0)

b[btop--]='\0';

}

void display()

{

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

printf("%d%c",a[i],b[i]);

}

void display1(char p[],int m) //Displays The Present Input String

{

int l;

printf("\t\t");

for(l=m;p[l]!='\0';l++)

printf("%c",p[l]);

printf("\n");

}

void error()

{

printf("Syntax Error");

}

void reduce(int p)

{

int len,k,ad;

char src,\*dest;

switch(p)

{

case 1:dest="E+T";

src='E';

break;

case 2:dest="T";

src='E';

break;

case 3:dest="T\*F";

src='T';

break;

case 4:dest="F";

src='T';

break;

case 5:dest="(E)";

src='F';

break;

case 6:dest="i";

src='F';

break;

default:dest="\0";

src='\0';

break;

} for(k=0;k<strlen(dest);k++)

{

pop();

popb();

}

pushb(src);

switch(src)

{

case 'E':ad=0;

break;

case 'T':ad=1;

break;

case 'F':ad=2;

break;

default: ad=-1;

break;

}

push(gotot[TOS()][ad]);

} int main()

{

int j,st,ic;

char ip[20]="\0",an;

// clrscr();

printf("Enter any String\n");

+

scanf("%s",ip);

push(0);

display();

printf("\t%s\n",ip);

for(j=0;ip[j]!='\0';)

{

st=TOS();

an=ip[j];

if(an>='a'&&an<='z') ic=0;

else if(an=='+') ic=1;

else if(an=='\*') ic=2;

else if(an=='(') ic=3;

else if(an==')') ic=4;

else if(an=='$') ic=5;

else {

error();

break;

}

if(axn[st][ic][0]==100)

{

pushb(an);

push(axn[st][ic][1]);

display();

j++;

display1(ip,j);

}

if(axn[st][ic][0]==101)

{

reduce(axn[st][ic][1]);

display();

display1(ip,j);

}

if(axn[st][ic][1]==102)

{

printf("Given String is accepted \n");

// getch();

break;

}

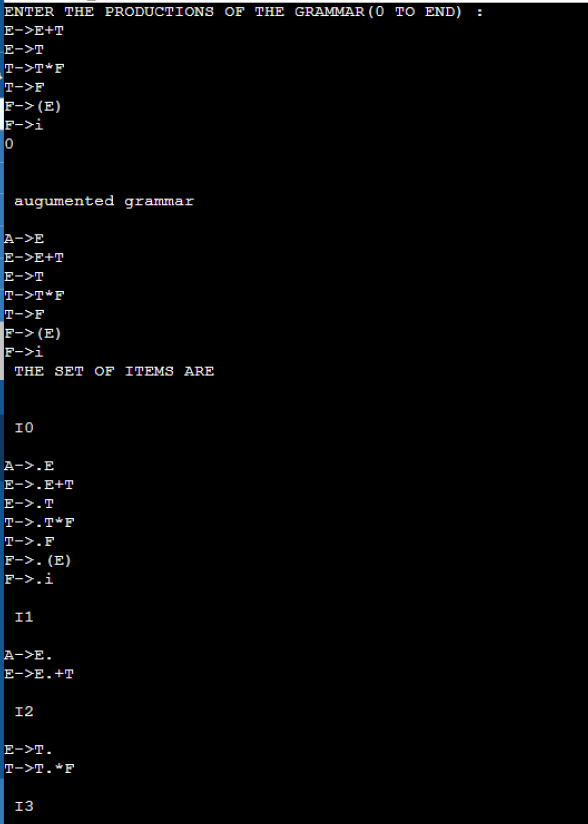
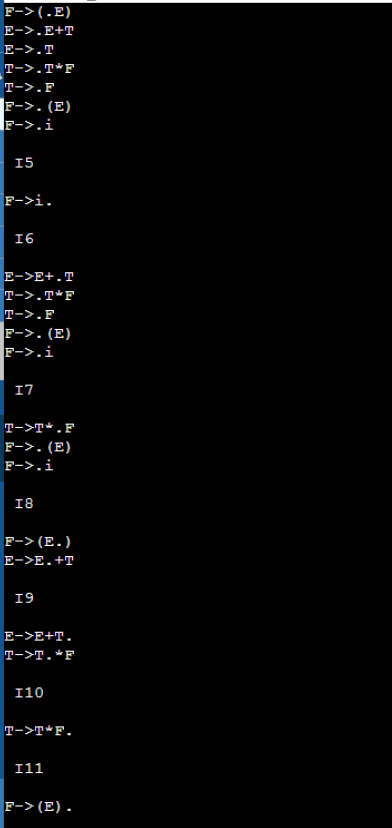
}

return 0;

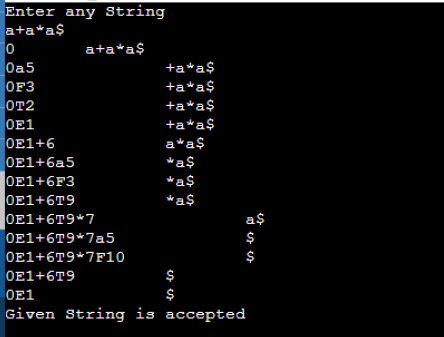
}

**LR & SLR OUTPUT:**

**LR**

** **

**SLR**

****

**RESULT:** Hence, the implementation of LR & SLR is successfully done.

**CD LAB 10 – LEADING & TRAILING**

**RAJAT KUMAR**

**RA1911003010652**

**CODE:**

#include<iostream>

#include<conio.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

using namespace std;

int vars,terms,i,j,k,m,rep,count,temp=-1;

char var[10],term[10],lead[10][10],trail[10][10];

struct grammar

{

int prodno;

char lhs,rhs[20][20];

}gram[50];

void get()

{

cout<<"\n LEADING AND TRAILING\n";

cout<<"\nEnter the no. of variables : ";

cin>>vars;

cout<<"\nEnter the variables : \n";

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

{

cin>>gram[i].lhs;

var[i]=gram[i].lhs;

}

cout<<"\nEnter the no. of terminals : ";

cin>>terms;

cout<<"\nEnter the terminals : ";

for(j=0;j<terms;j++)

cin>>term[j];

cout<<"\n PRODUCTION DETAILS\n";

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

{

cout<<"\nEnter the no. of production of "<<gram[i].lhs<<":";

cin>>gram[i].prodno;

for(j=0;j<gram[i].prodno;j++)

{

cout<<gram[i].lhs<<"->";

cin>>gram[i].rhs[j];

}

}

}

void leading()

{

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

{

for(j=0;j<gram[i].prodno;j++)

{

for(k=0;k<terms;k++)

{

if(gram[i].rhs[j][0]==term[k])

lead[i][k]=1;

else

{

if(gram[i].rhs[j][1]==term[k])

lead[i][k]=1;

}

}

}

}

for(rep=0;rep<vars;rep++)

{

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

{

for(j=0;j<gram[i].prodno;j++)

{

for(m=1;m<vars;m++)

{

if(gram[i].rhs[j][0]==var[m])

{

temp=m;

goto out;

}

}

out:

for(k=0;k<terms;k++)

{

if(lead[temp][k]==1)

lead[i][k]=1;

}

}

}

}

}

void trailing()

{

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

{

for(j=0;j<gram[i].prodno;j++)

{

count=0;

while(gram[i].rhs[j][count]!='\x0')

count++;

for(k=0;k<terms;k++)

{

if(gram[i].rhs[j][count-1]==term[k])

trail[i][k]=1;

else

{

if(gram[i].rhs[j][count-2]==term[k])

trail[i][k]=1;

}

}

}

}

for(rep=0;rep<vars;rep++)

{

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

{

for(j=0;j<gram[i].prodno;j++)

{

count=0;

while(gram[i].rhs[j][count]!='\x0')

count++;

for(m=1;m<vars;m++)

{

if(gram[i].rhs[j][count-1]==var[m])

temp=m;

}

for(k=0;k<terms;k++)

{

if(trail[temp][k]==1)

trail[i][k]=1;

}

}

}

}

}

void display()

{

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

{

cout<<"\nLEADING("<<gram[i].lhs<<") = ";

for(j=0;j<terms;j++)

{

if(lead[i][j]==1)

cout<<term[j]<<",";

}

}

cout<<endl;

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

{

cout<<"\nTRAILING("<<gram[i].lhs<<") = ";

for(j=0;j<terms;j++)

{

if(trail[i][j]==1)

cout<<term[j]<<",";

}

}

}

int main()

{

get();

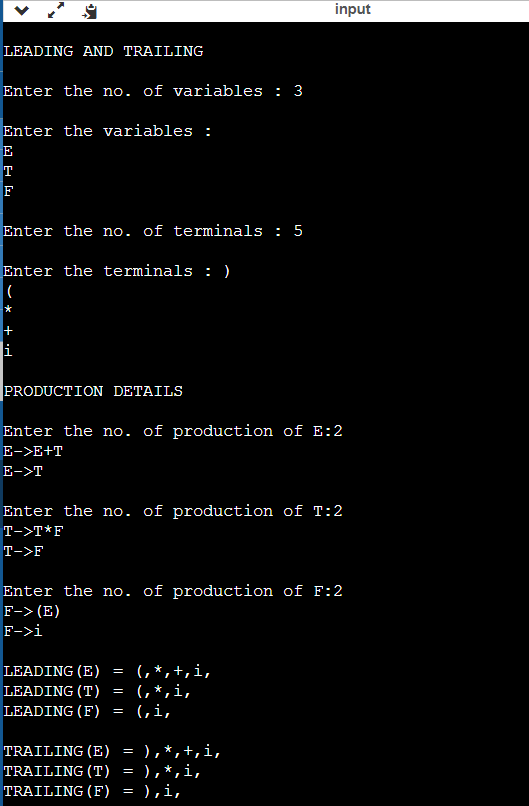
leading();

trailing();

display();

}

**OUTPUT:**



**RESULT:** Hence, the implementation of LEADING and TRAILING is successfully done.