Compiler Design Lab

NAME: Priyanka Srinivas
REGISTER NUMBER: RA1911026010014
SECTION: K1

Exercise 8: Computation of leading and trailing

AIM: To write a program to implement leading and trailing

INTRODUCTION:

What is leading?

If production is of form A → aα or A → Ba α where B is Non-terminal, and α can be any string, then the first terminal symbol on R.H.S is

Leading(A) = {a}

If production is of form A → Bα, if a is in LEADING (B), then a will also be in LEADING (A).

What is trailing?

If production is of form A→ αa or A → αaB where B is Non-terminal, and α can be any string, then,

Trailing(A) = {a}

If production is of form A → αB. If a is in TRAILING (B), then a will be in TRAILING (A).

ALGORITHM:

1. Start the program.
2. For each non-terminal A and terminal a: L [A, a] = false;
3. For each production of form **A → aα** or **A → Baα**: Install (A, a);
4. While the stack not empty:
   (a) Pop top pair (B, a) from stack;
   (b) For each production of form A → B α -> Install (A, a);
5. Display the productions without left recursion.
6. Stop the program.

PROGRAM:

```
#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<<"\nLEADING 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<<"\nPRODUCTION 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')
```

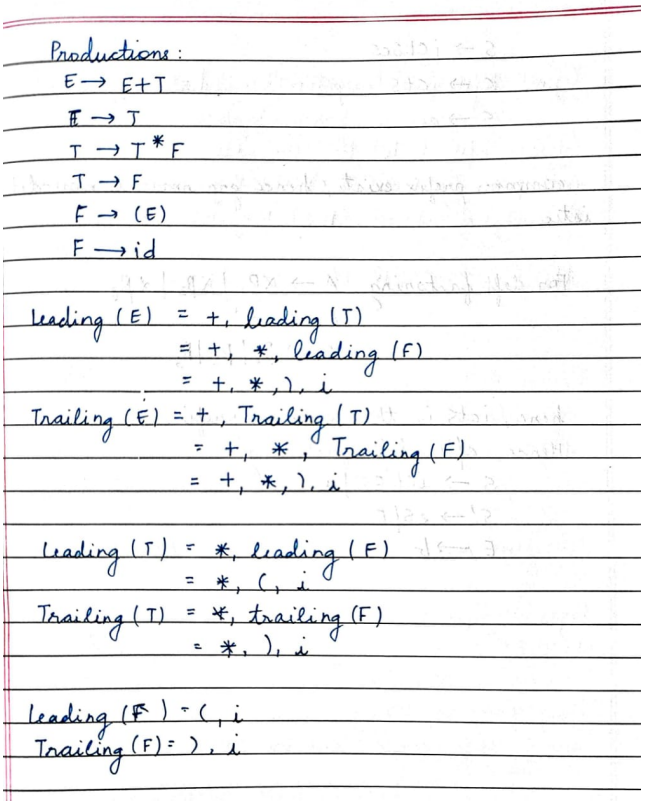
```
                    count++;
               for(k=0;k<terms;k++)
               {
                    if(gram[i].rhs[j][count-1]==term[k])
                         trail[i][k]=1;
                    else
                    {
                         if(gram[i].rhs[j][count-2]==term[k])
                              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();

}
```

INPUT:

E->E+T
E->T
T->T*F
T->F
F->(E)
F->i

Manual Output:



Productions:

E → E+T

~~E~~ → T

T → T*F

T → F

F → (E)

F → id

Leading (E) = +, leading (T)
= +, *, leading (F)
= +, *,), i

Trailing (E) = +, Trailing (T)
= +, *, Trailing (F)
= +, *,), i

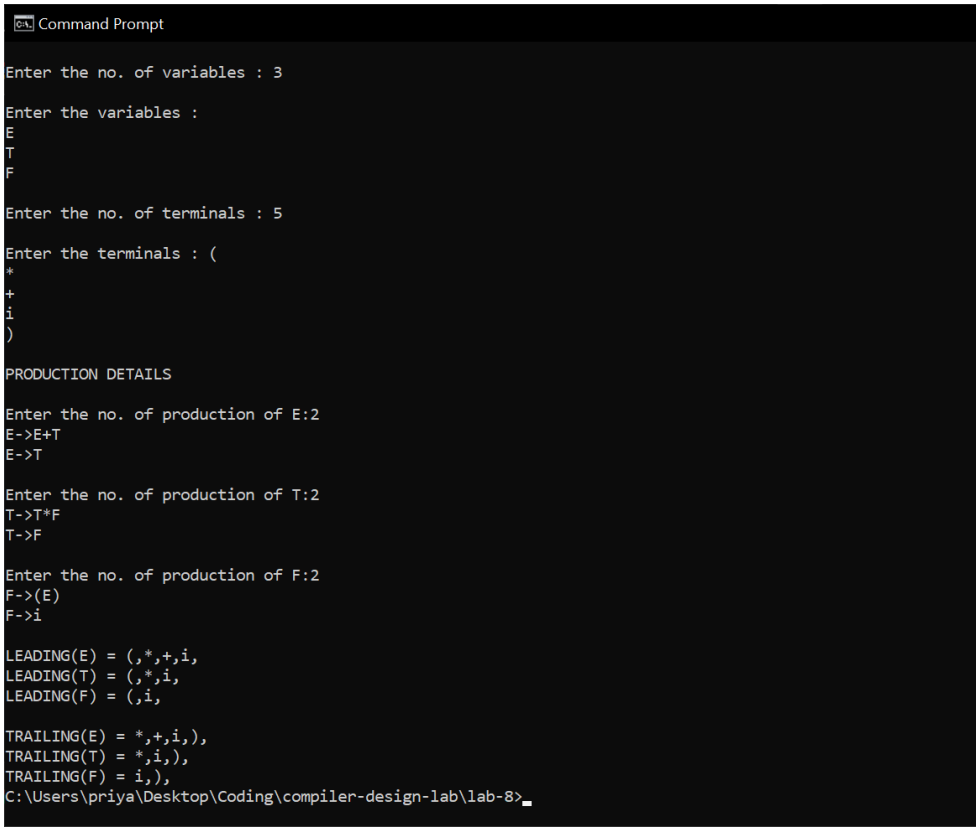
Leading (T) = *, leading (F)
= *, (, i

Trailing (T) = *, trailing (F)
= *,), i

Leading (~~F~~) = (, i

Trailing (F) =), i

System Output:



RESULT:

For a grammar, leading and trailing were computed.