**COMPILER DESIGN PRACTICAL LAB-8**

**Computation of LEADING AND TRAILING**

SANDRA MARIA TONY RA1911026010045 K1

**AIM:** A program for construction of **Computation of LEADING AND TRAILING.**

**Algorithm:**

For leading terminal:

1. Start

2. For each nonterminal A and terminal a do L(A,a):= false;

3. For each production of the form A->a or A->B do

INSTALL(A,a);

4. While STACK not empty repeat step 5& 6

5. Pop top pair (B,a) from STACK;

6. For each production of the form A->B do

INSTALL(A,a)

7. Stop

For INSTALL(A,a)

1. Start

2. If L(A,a) not present do step 3 and 4. 3 . Make L(A,a)=True

4 . Push (A,a) onto stack 5 . Stop

For trailing Terminal

1. Start

2. For each non terminal A and terminal a do L(A,a):=false;

3. For each production of the form A->a(alpha) or A-> Ba(alpha) do INSTALL(A,a)

4. While STACK not empty repeat 5 and 6

5. Pop top pair from stack

6. For each production of the form A-> B(alpha) do INSTALL(A,a)

7. Stop

For INSTALL(A,a)

1. Start

2. If L[A,a] not present repeat step 3 and 4

3. Make L(A,a)=True

4. Push (A,a) onto stack

5. Stop

**CODE:**

#include <iostream>

using namespace std;

#include<string.h>

int nt,t,top=0;

char s[50],NT[10],T[10],st[50],l[10][10],tr[50][50];

int searchnt(char a)

{

int count=-1,i;

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

{

if(NT[i]==a)

return i;

}

return count;

}

int searchter(char a)

{

int count=-1,i;

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

{

if(T[i]==a)

return i;

}

return count;

}

void push(char a)

{

s[top]=a;

top++;

}

char pop()

{

top--;

return s[top];

}

void installl(int a,int b)

{

if(l[a][b]=='f')

{

l[a][b]='t';

push(T[b]);

push(NT[a]);

}

}

void installt(int a,int b)

{

if(tr[a][b]=='f')

{

tr[a][b]='t';

push(T[b]);

push(NT[a]);

}

}

int main()

{

int i,s,k,j,n;

char pr[30][30],b,c;

cout<<"Enter the no of productions:";

cin>>n;

cout<<"Enter the productions one by one\n";

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

cin>>pr[i];

nt=0;

t=0;

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

{

if((searchnt(pr[i][0]))==-1)

NT[nt++]=pr[i][0];

}

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

{

for(j=3;j<strlen(pr[i]);j++)

{

if(searchnt(pr[i][j])==-1)

{

if(searchter(pr[i][j])==-1)

T[t++]=pr[i][j];

}

}

}

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

{

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

l[i][j]='f';

}

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

{

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

tr[i][j]='f';

}

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

{

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

{

if(NT[(searchnt(pr[j][0]))]==NT[i])

{

if(searchter(pr[j][3])!=-1)

installl(searchnt(pr[j][0]),searchter(pr[j][3]));

else

{

for(k=3;k<strlen(pr[j]);k++)

{

if(searchnt(pr[j][k])==-1)

{

installl(searchnt(pr[j][0]),searchter(pr[j][k]));

break;

}}}}}}

while(top!=0)

{

b=pop();

c=pop();

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

{

if(pr[s][3]==b)

installl(searchnt(pr[s][0]),searchter(c));

}

}

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

{

cout<<"Leading["<<NT[i]<<"]"<<"\t{";

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

{

if(l[i][j]=='t')

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

}

cout<<"}\n";

}

top=0;

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

{

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

{

if(NT[searchnt(pr[j][0])]==NT[i])

{

if(searchter(pr[j][strlen(pr[j])-1])!=-1)

installt(searchnt(pr[j][0]),searchter(pr[j][strlen(pr[j])-1]));

else

{

for(k=(strlen(pr[j])-1);k>=3;k--)

{

if(searchnt(pr[j][k])==-1)

{

installt(searchnt(pr[j][0]),searchter(pr[j][k]));

break;

}}}}}}

while(top!=0)

{

b=pop();

c=pop();

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

{

if(pr[s][3]==b)

installt(searchnt(pr[s][0]),searchter(c));

}

}

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

{

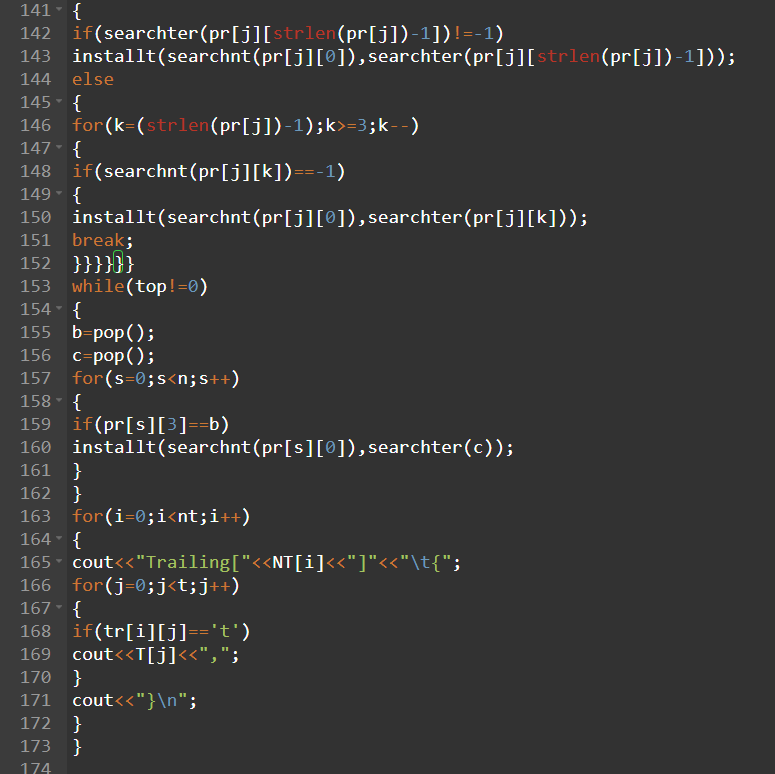
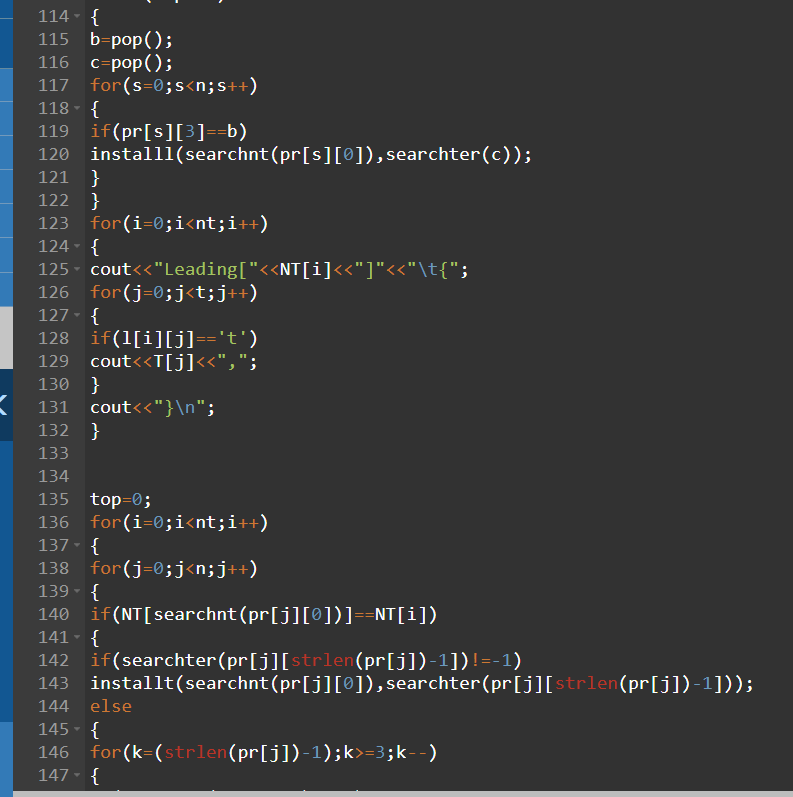
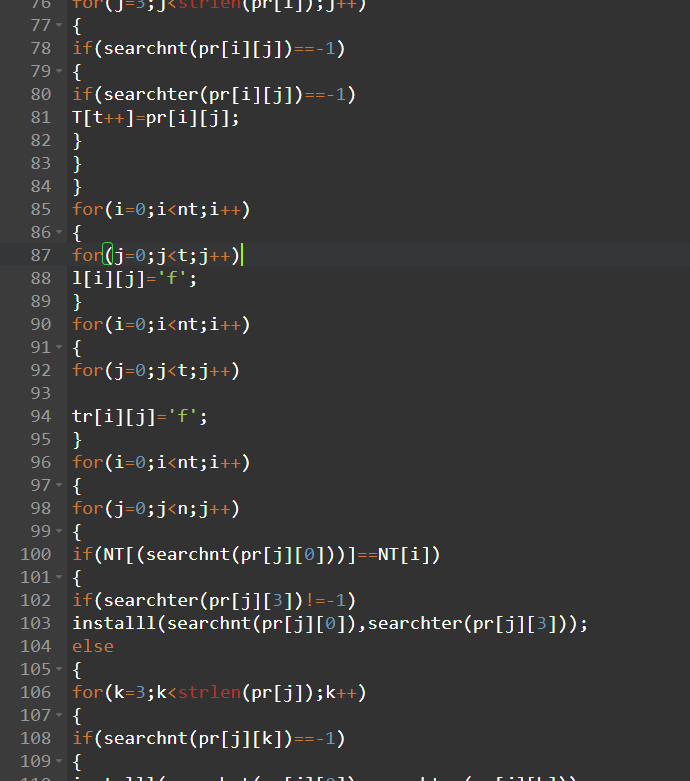
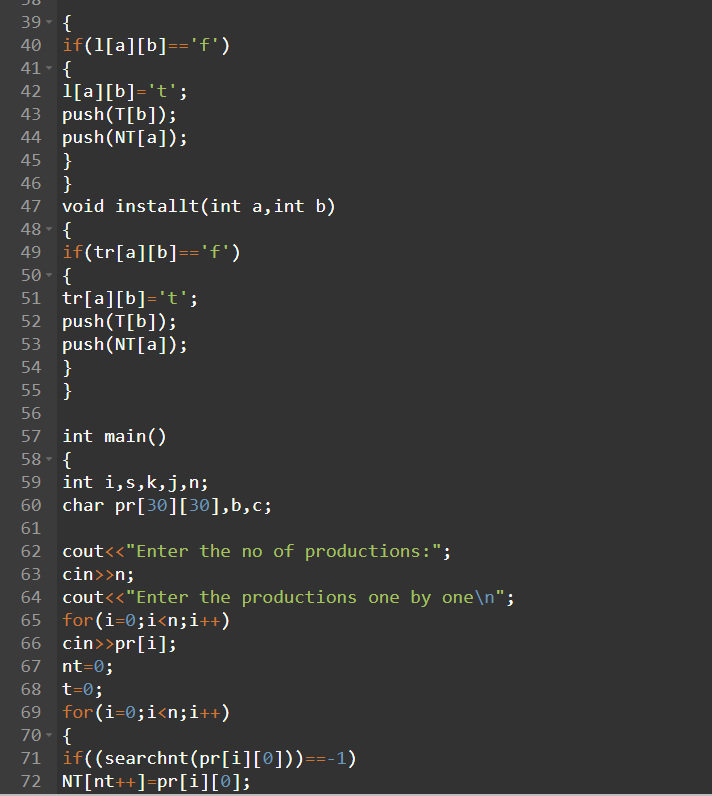
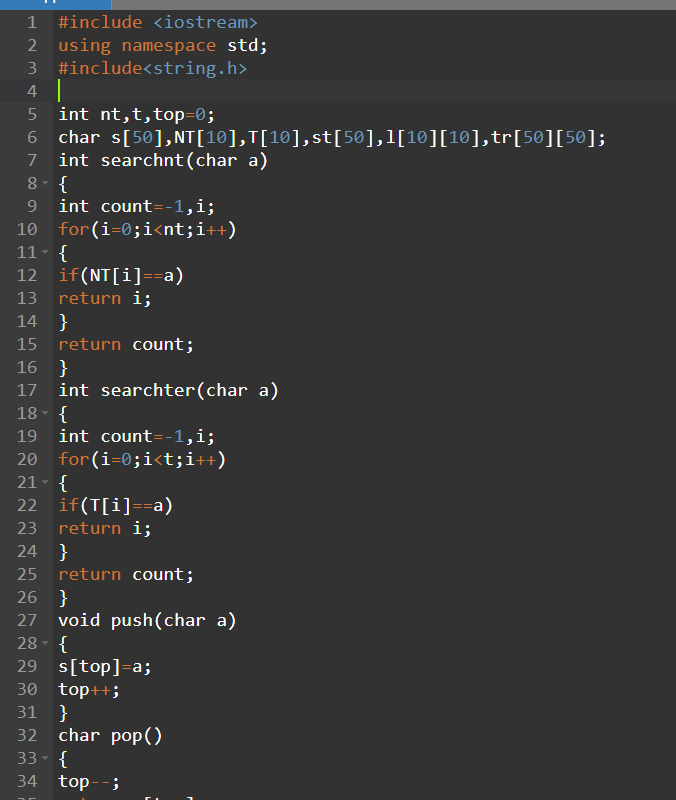
cout<<"Trailing["<<NT[i]<<"]"<<"\t{";

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

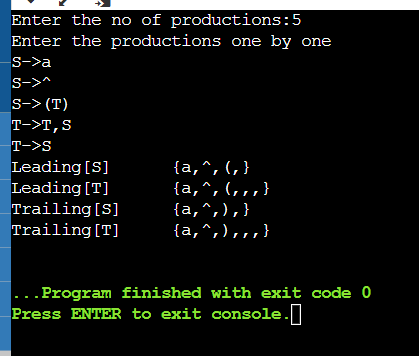
{if(tr[i][j]=='t')

cout<<T[j]<<",";}

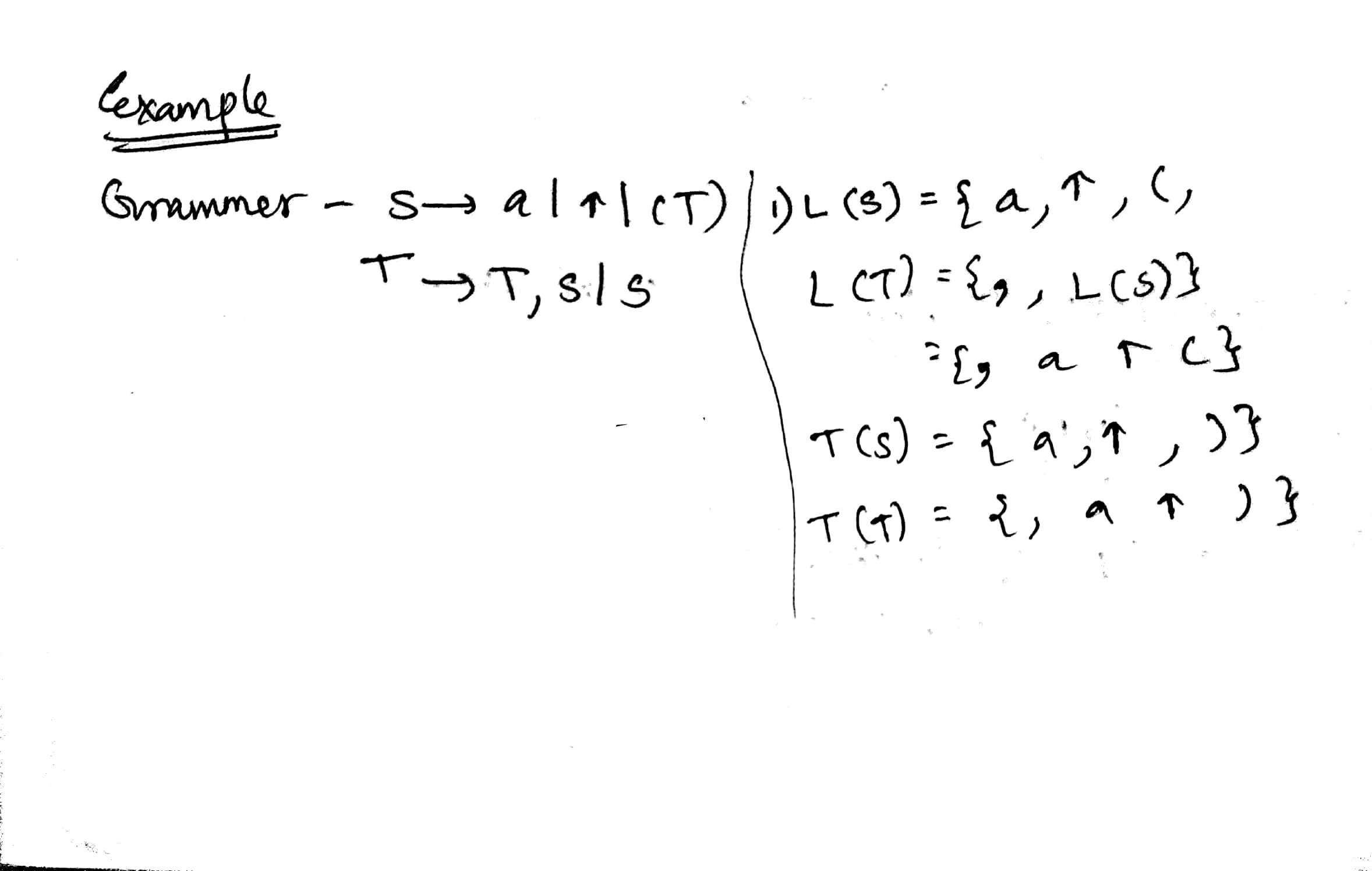
cout<<"}\n";}}



**Output:**

****

**MANUAL CALCULATION:**



kjhkjh

**RESULT:** The code for construction of **Computation of LEADING AND TRAILING** produced same output as the result from manual calculation thus the program was executed successfully.