ELIMINATION OF LEFT RECURSION

EX. NO. 4(a)

Parth Langalia(RA2011033010033) DATE

: 14/02/23

AIM: A program for Elimination of Left Recursion.

ALGORITHM:

1. Start the program.
2. Initialize the arrays for taking input from the user.
3. Prompt the user to input the no. of non-terminals having left recursion and no. of productions for these non-terminals.
4. Prompt the user to input the production for non-terminals.
5. Eliminate left recursion using the following rules:-

   A->Aα1| Aα2 | . . . . . |Aαm

   A->β1| β2| . . . . .| βn Then

   replace it by

   A-> βi A' i=1,2,3,…..m

   A'-> αj A' j=1,2,3,…..n

   A'-> ε
6. After eliminating the left recursion by applying these rules, display the productions without left recursion.
7. Stop.

CODE:

```
#include <iostream>
#include <vector>
#include <string> using
namespace std;
```

```
int main()
{      int
n;
   cout<<"\nEnter number of non terminals: ";
cin>>n;
   cout<<"\nEnter non terminals one by one: ";
int i;
   vector<string> nonter(n);
vector<int> leftrecr(n,0);     for(i=0;i<n;++i)
{
          cout<<"\nNon terminal "<<i+1<<" : ";
cin>>nonter[i];
   }
   vector<vector<string> > prod;
cout<<"\nEnter '^' for null";     for(i=0;i<n;++i)
{         cout<<"\nNumber of "<<nonter[i]<<"
productions: ";         int k;         cin>>k;
       int j;
       cout<<"\nOne by one enter all "<<nonter[i]<<" productions";
vector<string> temp(k);         for(j=0;j<k;++j) {
          cout<<"\nRHS of production "<<j+1<<": ";
string abc;
cin>>abc;            temp[j]=abc;

if(nonter[i].length()<=abc.length()&&nonter[i].compare(abc.substr(0,nonter[i].length()))==0)
leftrecr[i]=1;
       }
       prod.push_back(temp);
   }
for(i=0;i<n;++i) {
cout<<leftrecr[i];
```

```
   }
for(i=0;i<n;++i) {
if(leftrecr[i]==0)            continue;
int j;
       nonter.push_back(nonter[i]+"'");
vector<string> temp;         for(j=0;j<prod[i].size();++j)
{

if(nonter[i].length()<=prod[i][j].length()&&nonter[i].compare(prod[i][j].substr(0,nonter[i].lengt
h
()))==0) {
               string
abc=prod[i][j].substr(nonter[i].length(),prod[i][j].length()-nonter[i].length())+nonter[i]+"'";
temp.push_back(abc);                 prod[i].erase(prod[i].begin()+j);
               --j;
}             else
{
prod[i][j]+=nonter[i]+"'";
           }
       }
       temp.push_back("^");
prod.push_back(temp);
   }
   cout<<"\n\n";      cout<<"\nNew set of
non-terminals: ";
for(i=0;i<nonter.size();++i)
cout<<nonter[i]<<" ";     cout<<"\n\nNew
set of productions: ";
for(i=0;i<nonter.size();++i) {
```

```
        int j;
for(j=0;j<prod[i].size();++j) {
cout<<"\n"<<nonter[i]<<" -> "<<prod[i][j];
        }     }
return 0;  }
```

OUTPUT :

```
Enter number of non terminals: 3

Enter non terminals one by one:
Non terminal 1 : E

Non terminal 2 : T

Non terminal 3 : F

Enter '^' for null
Number of E productions: 2

One by one enter all E productions
RHS of production 1: E+T

RHS of production 2: T

Number of T productions: 2

One by one enter all T productions
RHS of production 1: T*F

RHS of production 2: F

Number of F productions: 2

One by one enter all F productions
RHS of production 1: (E)

RHS of production 2: i
110


New set of non-terminals: E T F E' T'

New set of productions:
E -> TE'
T -> FT'
F -> (E)
F -> i
E' -> +TE'
E' -> ^
T' -> *FT'
T' -> ^
```

RESULT :

A program for Elimination of Left Recursion was run successfully.

LEFT FACTORING

EX. NO. 4(b)

Parth Langalia(RA2011033010033)

AIM : A program for implementation Of Left Factoring

ALGORITHM :

1. Start
2. Ask the user to enter the set of productions
3. Check for common symbols in the given set of productions by comparing with: A->aB1|aB2
4. If found, replace the particular productions with:

   A->aA'

   A'->B1 | B2|ε
5. Display the output
6. Exit

CODE :

```
#include <iostream>
#include <math.h>
#include <vector>
#include <string>
#include <stdlib.h> using
namespace std;

int main() {
   cout<<"\nEnter number of productions: ";
int p;     cin>>p;
   vector<string> prodleft(p),prodright(p);
cout<<"\nEnter productions one by one: ";     int i;
```

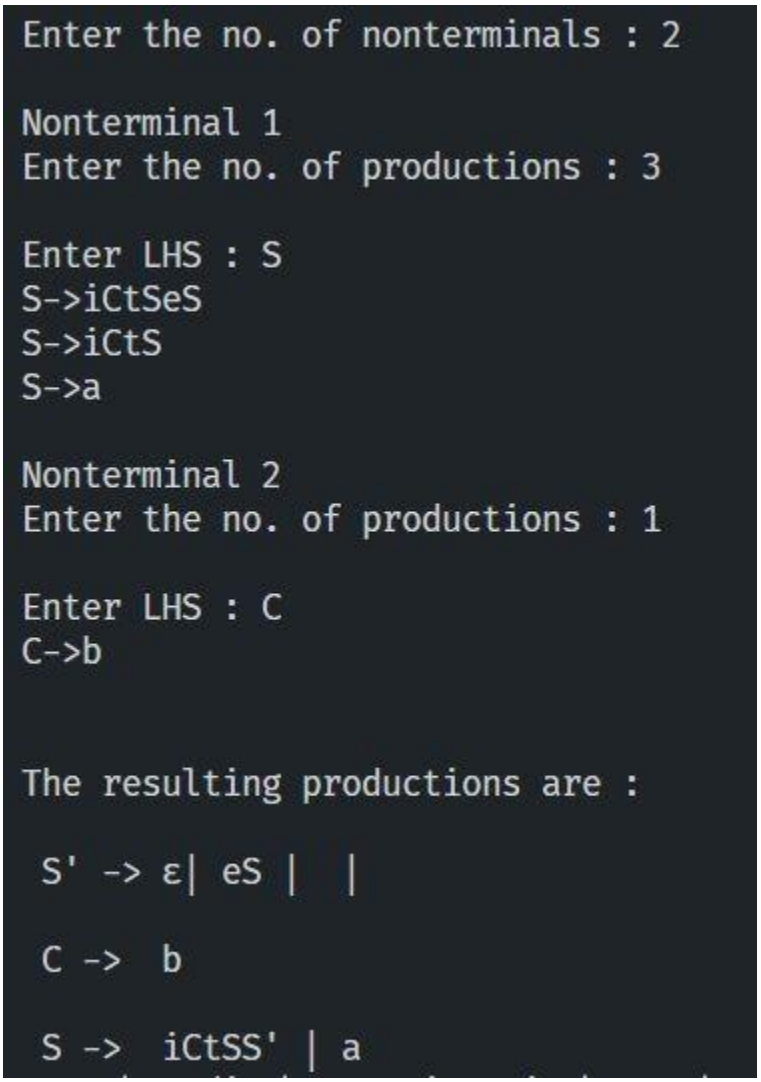
```
   for(i=0;i<p;++i) {          cout<<"\nLeft of
production "<<i+1<<": ";        cin>>prodleft[i];
       cout<<"\nRight of production "<<i+1<<": ";
cin>>prodright[i];
   }     int j;     int e=1;
for(i=0;i<p;++i) {
for(j=i+1;j<p;++j) {
if(prodleft[j]==prodleft[i]) {
int k=0;                string com="";

while(k<prodright[i].length()&&k<prodright[j].length()&&prodright[i][k]==prodright[j][k]) {
com+=prodright[i][k];
                 ++k;
}               if(k==0)
continue;                char*
buffer;
              string comleft=prodleft[i];
if(k==prodright[i].length()) {
prodleft[i]+=string(itoa(e,buffer,10));
prodleft[j]+=string(itoa(e,buffer,10));                    prodright[i]="^";
                 prodright[j]=prodright[j].substr(k,prodright[j].length()-k);
              }
              else if(k==prodright[j].length()) {
                 prodleft[i]+=string(itoa(e,buffer,10));
prodleft[j]+=string(itoa(e,buffer,10));                    prodright[j]="^";
                 prodright[i]=prodright[i].substr(k,prodright[i].length()-k);
              }
else {
                 prodleft[i]+=string(itoa(e,buffer,10));
prodleft[j]+=string(itoa(e,buffer,10));
                 prodright[j]=prodright[j].substr(k,prodright[j].length()-k);
```

```
prodright[i]=prodright[i].substr(k,prodright[i].length()-k);
            }               int
l;
            for(l=j+1;l<p;++l) {

if(comleft==prodleft[l]&&com==prodright[l].substr(0,fmin(k,prodright[l].length()))) {
prodleft[l]+=string(itoa(e,buffer,10));
prodright[l]=prodright[l].substr(k,prodright[l].length()-k);
                }
            }
prodleft.push_back(comleft);
prodright.push_back(com+prodleft[i]);
            ++p;
            ++e;
          }
       }
   }
   cout<<"\n\nNew productions";
for(i=0;i<p;++i) {
       cout<<"\n"<<prodleft[i]<<"->"<<prodright[i];
   }     return
0; }
```

OUTPUT :



RESULT : A program for implementation Of Left Factoring was compiled and run successfully