VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

COMPILER DESIGN LAB
LEFT RECURSION & LEFT FACTORING

Name of the Student: SreeDananjay S
Registration Number: 21BAI1807
Slot: L31+L32
Course Code: BCSE307P
Programme: Bachelor of Technology in Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning
School: School of Computer Science and Engineering(SCOPE)

1. Write a program to eliminate left recursion.

CODE:

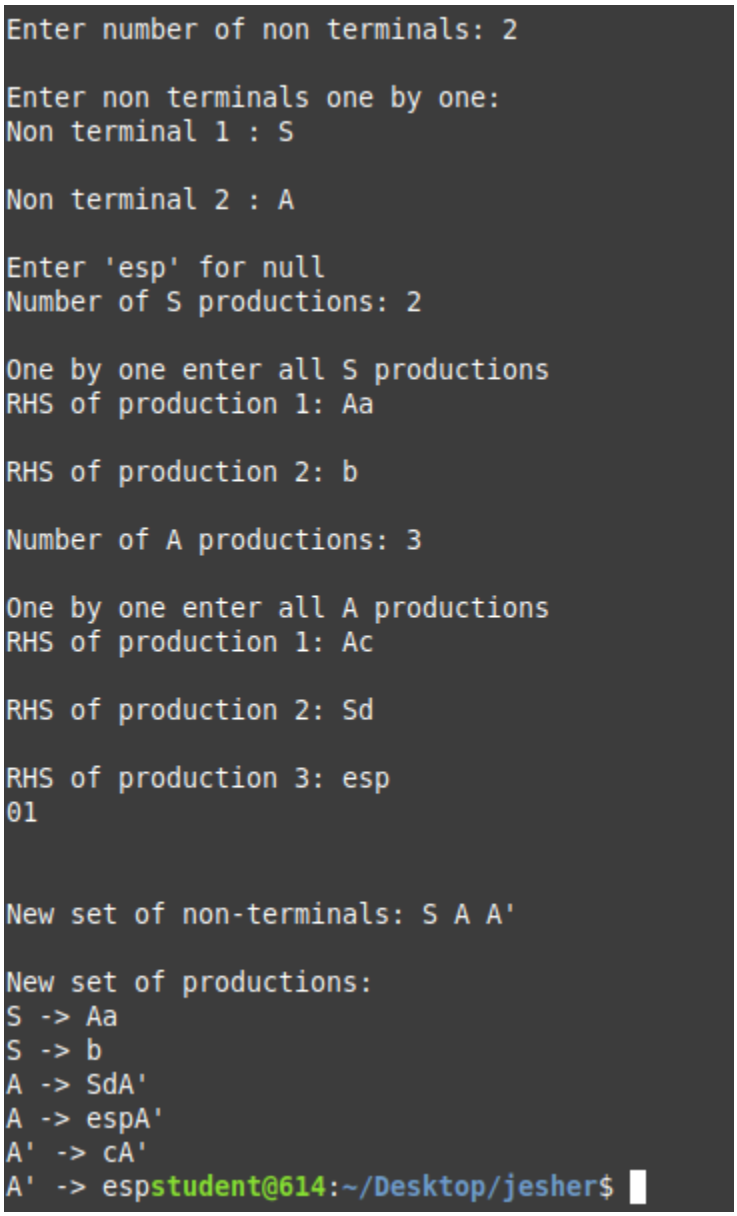
```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

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 'esp' 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].length())) == 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("esp");
        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: 2
Enter non terminals one by one:
Non terminal 1 : S
Non terminal 2 : A
Enter 'esp' for null
Number of S productions: 2
One by one enter all S productions
RHS of production 1: Aa
RHS of production 2: b
Number of A productions: 3
One by one enter all A productions
RHS of production 1: Ac
RHS of production 2: Sd
RHS of production 3: esp
01
New set of non-terminals: S A A'
New set of productions:
S -> Aa
S -> b
A -> SdA'
A -> espA'
A' -> cA'
A' -> espstudent@614:~/Desktop/jesher$

2.Write a Program to eliminate Left-factoring

CODE:

```
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
void main()
{char ch,lhs[20][20],rhs[20][20][20],temp[20],temp1[20];
        int n,n1,count[20],x,y,i,j,k,c[20];
        printf("\nEnter the no. of nonterminals : ");
        scanf("%d",&n);
        n1=n;
        for(i=0;i<n;i++)
        {
                printf("\nNonterminal %d \nEnter the no. of productions : ",i+1);
                scanf("%d",&c[i]);
                printf("\nEnter LHS : ");
                scanf("%s",lhs[i]);
                for(j=0;j<c[i];j++)
                {
                        printf("%s->",lhs[i]);
                        scanf("%s",rhs[i][j]);
                }}
        for(i=0;i<n;i++)
        {
                count[i]=1;
                while(memcmp(rhs[i][0],rhs[i][1],count[i])==0)
                        count[i]++;
        }for(i=0;i<n;i++)
        {
                count[i]--;
                if(count[i]>0)
                {
                        strcpy(lhs[n1],lhs[i]);
                        strcat(lhs[i],"'");
                        for(k=0;k<count[i];k++)
                                temp1[k] = rhs[i][0][k];
                        temp1[k++] = '\0';
                        for(j=0;j<c[i];j++)
                        {
                                for(k=count[i],x=0;k<strlen(rhs[i][j]);x++,k++)
                                        temp[x] = rhs[i][j][k];
                                temp[x++] = '\0';
                                if(strlen(rhs[i][j])==1)
```

```
                                        strcpy(rhs[n1][1],rhs[i][j]);
                                strcpy(rhs[i][j],temp);
                        }
                        c[n1]=2;
                        strcpy(rhs[n1][0],temp1);
                        strcat(rhs[n1][0],lhs[n1]);
                        strcat(rhs[n1][0],"'");
                        n1++;}
        }
        printf("\n\nThe resulting productions are : \n");
        for(i=0;i<n1;i++)
        {
                if(i==0)
                        printf("\n %s -> %c|",lhs[i],(char)238);
                else
                        printf("\n %s -> ",lhs[i]);
                for(j=0;j<c[i];j++)
                {
                        printf(" %s ",rhs[i][j]);
                        if((j+1)!=c[i])
                                printf("|");
                }
                printf("\b\b\b\n");}}
```

OUTPUTS:

```
Enter the no. of nonterminals : 2

Nonterminal 1
Enter the no. of productions : 3

Enter LHS : S
S->iEtS
S->iEtSeS
S->a

Nonterminal 2
Enter the no. of productions : 1

Enter LHS : E
E->b


The resulting productions are :

 S' -> ?|  | eS |

 E ->  b

 S ->  iEtSS' | a
```