**ABHIJIT DASH**

**RA1811031010112**

**ELIMINATION OF LEFT FACTORING**

**AIM:-**

To write a program to eliminate left factoring from a grammar.

**CODE:-**

#include<iostream>

#include<string>

using namespace std;

int main()

{ string ip,op1,op2,temp;

int sizes[10] = {};

char c;

int n,j,l;

cout<<"Enter the Parent Non-Terminal : ";

cin>>c;

ip.push\_back(c);

op1 += ip + "\'->";

op2 += ip + "\'\'->";;

ip += "->";

cout<<"Enter the number of productions : ";

cin>>n;

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

{

cout<<"Enter Production "<<i+1<<" : ";

cin>>temp;

sizes[i] = temp.size();

ip+=temp;

if(i!=n-1)

ip += "|";

}

cout<<"Production Rule : "<<ip<<endl;

char x = ip[3];

for(int i=0,k=3;i<n;i++)

{

if(x == ip[k])

{

if(ip[k+1] == '|')

{

op1 += "#";

ip.insert(k+1,1,ip[0]);

ip.insert(k+2,1,'\'');

k+=4;

}

else

{

op1 += "|" + ip.substr(k+1,sizes[i]-1);

ip.erase(k-1,sizes[i]+1);

}

}

else

{

while(ip[k++]!='|');

}

}

char y = op1[6];

for(int i=0,k=6;i<n-1;i++)

{

if(y == op1[k])

{

if(op1[k+1] == '|')

{

op2 += "#";

op1.insert(k+1,1,op1[0]);

op1.insert(k+2,2,'\'');

k+=5;

}

else

{

temp.clear();

for(int s=k+1;s<op1.length();s++)

temp.push\_back(op1[s]);

op2 += "|" + temp;

op1.erase(k-1,temp.length()+2);

}

}

}

op2.erase(op2.size()-1);

cout<<"After Left Factoring : "<<endl;

cout<<ip<<endl;

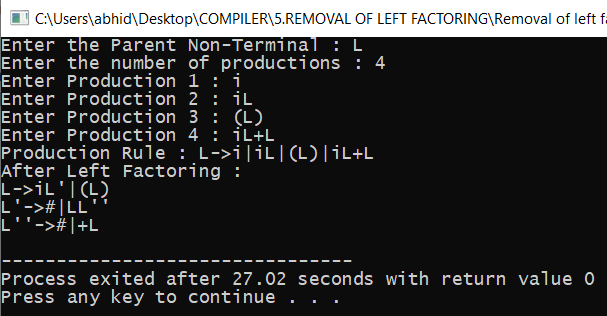
cout<<op1<<endl;

cout<<op2<<endl;

return 0;

}

**OUTPUT SCREENSHOTS:-**



**RESULT:-**

The code was run successfully and the output is produced.