**Experiment 5**

**Aim:** Write a program to implement left recursion removal algorithm.

**Input of program**- Grammar with possibly direct or indirect recursion.

**Output of program**- Equivalent grammar without the recursion problem.

**Tools:** g++ compiler, Text editor.

**C++ Code:-**

#include <iostream>

#include <stdlib.h>

#include <cstring>

#include <fstream>

using namespace std;

int main(int argc, char \*argv[])

{

char ch[255][255],alpha[255][255],beta[255][255];

int len[20] = {0};

int n = 0;

cout<<"Enter the number of production:- "<<endl;

cin>>n;

int j = 0;

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

{

cin>>ch[i];

}

int a=0,a1=-1,b=0,b1=-1;

char nt[n];

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

{

int index = 3;

nt[i] = ch[i][0];

a1=-1;b1=-1;

while(ch[i][index] != 0)

{

if(nt[i] == ch[i][index])

{

index++;

while(ch[i][index]!=0 && ch[i][index]!='|')

{

a1++;

alpha[a][a1] = ch[i][index];

cout<<"Alpha: "<<alpha[a][a1]<<" "<<a<<" "<<a1<<endl;

index++;

}

if(ch[i][index] == '|')

{

a1++;

alpha[a][a1] = ch[i][index];

cout<<"Alpha: "<<alpha[a][a1]<<" "<<a<<" "<<a1<<endl;

}

}

else

{

while(ch[i][index]!=0 && ch[i][index]!='|')

{

b1++;

beta[b][b1] = ch[i][index];

cout<<"Beta: "<<beta[b][b1]<<" "<<b<<" "<<b1<<endl;

index++;

}

if(ch[i][index] == '|')

{

b1++;

beta[b][b1] = ch[i][index];

cout<<"Beta: "<<beta[b][b1]<<" "<<b<<" "<<b1<<endl;

}

}

index++;

}

a++;b++;

}

cout<<"After Removing left recursion:\n";

int k=0,l=0;

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

{

k=0;l=0;

cout<<nt[i]<<"->";

while(beta[i][k] != 0)

{

while(beta[i][k]!=0 && beta[i][k]!='|')

{

cout<<beta[i][k];

k++;

}

cout<<nt[i]<<"'";

cout<<beta[i][k];

k++;

}

cout<<"\n"<<nt[i]<<"'"<<"->";

while(alpha[i][l] != 0)

{

while(alpha[i][l]!=0 && alpha[i][l]!='|')

{

cout<<alpha[i][l];

l++;

}

cout<<nt[i]<<"'";

cout<<"|";

l++;

}

cout<<"^\n";

}

return 0;

}

**Output:-**



