**Experiment: 01**

**Objective:** It is required to write a program to implement left factoring

**Theory:** Left Factoring is a process to remove the common prefix problem where parser gets confused when single input can generate different productions. As a single input can lead to many productions, it has many options to go so the parser becomes confused. It is also called Non-deterministic grammar and elimination of NDG can be called Left factoring.

Taking the common symbol with the prime of the nonterminal then the prime of non terminal will generate the rest of the production. Left factoring is a grammar transformation that is useful for producing grammar suitable for predictive parsing. The basic idea is that when it is not clear which of two alternative productions to use to expand a non-terminal A, we may be able to rewrite the A-productions to defer the decision until we have seen enough of the input to make the right choice.

**Source Code:**

#include <iostream>

#include <string>

using namespace std;

int main(){

char c;

int n;

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

cin>>c;

cout<<"Enter the Number of Productions :";

cin>>n;

string arr[n];

string common="";

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

cout<<"Enter the production "<<i+1<<":";

cin>>arr[i];

}

cout<<"The Entered Grammer is :"<<c<<" -> ";

for(int i=0;i<n-1;i++){

cout<<arr[i]<<"| ";

}

cout<<arr[n-1]<<endl;

for(int i=0;i<arr[0].size();i++){

bool check=true;

for(int j=1;j<n;j++){

if(arr[j][i]!=arr[0][i])

check=false;

break;

}

if(check==true) common+=arr[0][i];

}

int s=common.size();

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

arr[i].erase(0,s);

if(arr[i]=="")

arr[i]="e";

}

cout<<"\nAfter Left Factoring :"<<endl;

cout<<c<<" -> "<<common<<c<<"'"<<endl;

cout<<c<<"' -> ";

for(int i=0;i<n-1;i++){

cout<<arr[i]<<"| ";

}

cout<<arr[n-1];

}

**Output:**

