Practical 4

Name: Rajatkumar Patel
Roll No.: 18BCE191

Aim

To implement the Left most derivation removal algorithm.

Code

```
#include<bits/stdc++.h>
#define FAST ios_base::sync_with_stdio(false);cin.tie();cout.tie();
#define FILE_READ_IN freopen("input4.txt","r",stdin);
#define FILE_READ_OUT freopen("output.txt","w",stdout);
#define EPSILON '!'

using namespace std;

typedef long long ll;


void replaceAll(map<char,vector<string>>& prods,char left,char
right){
   vector<string> newprods;
   for(string &s:prods[left]){
       if(s[0]==right){
           for(string &x:prods[right]){
               newprods.push_back(x+s.substr(1,s.length()-1));
           }
       }
       else{
           newprods.push_back(s);
       }
   }
   prods[left]=newprods;
}
void removeImmediateLeftRecursion(map<char,vector<string>>&
prods,char curr,char &last){
```

```
    vector<string> left,nonleft;
    for(string &s:prods[curr]){
        if(s[0]==curr){
            left.push_back(s);
        }
        else nonleft.push_back(s);
    }

    if(left.empty()) return;

    for(string &s:nonleft){
       s.push_back(last);
    }
    for(string &s:left){
        string x = s.substr(1,s.length()-1);
        x+= curr;
        prods[last].push_back(x);
    }
    string ep = "";
    ep+= EPSILON;
    prods[last].push_back(ep);
    prods[curr] = nonleft;
    last--;
}
void eliminateLeftRecursion(map<char,vector<string>> & prods){
    char last = 'Z';
    for(auto i:prods){
        for(auto j:prods){
            if(j.first == i.first) break;
            vector<string> new_prods;
            // replace all production of j in i
            replaceAll(prods,i.first,j.first);
        }
        // A -> A c | d
        // A -> dA'
        // A'-> cA|epsilon
```

```
        removeImmediateLeftRecursion(prods,i.first,last);
    }
}
int main(){
  #ifndef ONLINE_JUDGE
     FILE_READ_IN
     FILE_READ_OUT
  #endif

    int n; cin>>n;
    map<char,vector<string>> prods;

    for(int i=0;i<n;i++){
        char nonterminal; cin>>nonterminal;
        string s; cin >> s;
        prods[nonterminal].push_back(s);
    }


    // epsilon is denoted by !

    eliminateLeftRecursion(prods);

    for(auto i:prods){
        cout<<i.first<<"->";
        for(int j=0;j<i.second.size();j++){
            cout << i.second[j];
            if(j+1 < i.second.size()) cout <<"|";
        }
        cout<<"\n";
    }


  return 0;
}
```

Input

input4.txt

```
6
A Bc
A Bd
A e
B Cf
C Ax
C efg
```

** ! = epsilon

Output

output.txt

```
A->Bc|Bd|e
B->Cf
C->exZ|efgZ
Z->fcxC|fdxC|!

```

** ! = epsilon

Conclusion

Implemented the algorithm to remove left recursion. The algorithm is capable of removing non-immediate left recursion as well as immediate left recursion.