Practical: 7

Aim: Implement a Program that remove left recursion on given grammar.

Program:

```
/*
    author: mr_bhishm
    created: 29-09-2020 18:20:17
    "Make it work, make it right, make it fast."
                                                    – Kent Beck
*/
#include<bits/stdc++.h>
using namespace std;
#define debug(x) cout<<#x<<" "<<x<<endl


// Only works for direct left recursion...
// i.e it works for production format like E->E+T | T


int main(){
    cout<<"Enter production rule: "<<endl;
    cout<<"[use CAPITAL for non-terminal and small case for terminal and ~ for
NULL]"<<endl;
    cout<<"Exmaple: S->aB without space"<<endl;

    string prod;
    cin>>prod;
    vector< string >  right;
    char left =  prod[0];
    string leftStr(1,left);
    string temp="";
    for(int i = 3 ; i < prod.length() ; i++){
        if(prod[i] != '|'){
            temp+=prod[i];
        }else{
            right.push_back(temp);
            temp = "";
        }
    }
    right.push_back(temp);
    temp = "";
    bool isLR = false;
    for(int i = 0 ; i < right.size() ; i++){
        if(left == right[i][0]){
            isLR = true;
            break;
        }
    }
```

```
    if(isLR == true){
        cout<<"Left Recurtion found!"<<endl;
        string ans = leftStr;
        ans+= "->";
        // debug(ans);
        string additional = leftStr ;
        additional+= "\'->";
        // debug(additional);
        for(int j = 0 ; j  < right.size() ; j++){
            if(right[j][0] != left){
                if(right[j][0] != '~'){
                    ans+=right[j]+leftStr + "\'|";
                }else{
                    ans+=leftStr + "\'|";
                }

            }else{
                for(int k = 1 ; k < right[j].length() ; k++){
                    additional+=right[j][k];
                }
                additional+=leftStr+"\'|";
            }
        }

        ans = ans.substr(0,ans.length()-1);
        additional+=(char)238;
        cout<<"----------------------------------------"<<endl;
        cout<<"Grammar After removing Left recursion"<<endl;
        cout<<"----------------------------------------"<<endl;
        cout<<ans<<endl;
        cout<<additional<<endl;
    }else{
        cout<<"No left recursion found!"<<endl;
        cout<<"----------------------------------------"<<endl;
        cout<<"No change in grammar"<<endl;
        cout<<"----------------------------------------"<<endl;
        cout<<prod<<endl;
    }

}
```

Output:

```
(base) PS D:\DLP_lab\Practical_7> g++ .\remove_LR.cpp
(base) PS D:\DLP_lab\Practical_7> .\a.exe
Enter production rule:
[use CAPITAL for non-terminal and small case for terminal and ~ for NULL]
Exmaple: S->aB without space
A->Abc|d|ef
Left Recurtion found!
--------------------------------------
Grammar After removing Left recursion
--------------------------------------
A->dA'|efA'
A'->bcA'|ε
```

Conclusion: From this practical I have learnt about how to remove left recursion from given grammar.