Practical: 8

Aim: Implement a program that performs left factoring on given grammar.

Program:

```
/*
    author: mr_bhishm
    created: 30-09-2020 16:37:46
    "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 work for single line one production
// i.e it works for production format like A->xab|xc


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_side;
    char left =  prod[0];
    string leftStr(1,left);
    string t="";
    for(int i = 3 ; i < prod.length() ; i++){
        if(prod[i] != '|'){
            t+=prod[i];
        }else{
            right_side.push_back(t);
            t = "";
        }
    }
    right_side.push_back(t);
    t = "";

    vector< string > result,temp;
    char c = right_side[0][0];
    for(int i  = 0 ; i < right_side.size() ; i++){
        if(c!= right_side[i][0]){
            result.push_back(leftStr+"->"+right_side[i]);
        }else{
            temp.push_back(right_side[i]);
```

```
        }
    }
    int len = temp[0].length();
    int smallest = 0;

    for(int i = 1 ; i < temp.size() ; i++){
        if(temp[i].length() < len){
            len = temp[i].length();
            smallest = i;
        }
    }
    // debug(temp[smallest].length());
    string str="";
    int ptr = 0;
    for(int i = 0; i < temp[smallest].length(); i++){
        char c = temp[smallest][i];
        // debug(c);
        int flag = 0;
        for(int j=0;j < temp.size();j++){
            if(temp[j][i] == c){
                // whole smallest stirng matches...
                if(j == temp.size()-1 && temp[smallest].length()-1 == i){
                    str = temp[smallest].substr(0,i+1);
                    flag = 1;
                    ptr = i+1;
                    break;
                }
                continue;
            }else{
                // debug(i);
                str = temp[smallest].substr(0,i);
                flag = 1;
                ptr = i;
                break;
            }
        }
        if(flag == 1)
            break;
    }

    // debug(str);
    result.push_back(leftStr+"->"+str+leftStr+"'");
    string additional = leftStr+"'->";
    for(int i=0;i<temp.size();i++){
        if(ptr==temp[i].length()){
            additional+= (char)238;
            additional+="|";
        }else{
            additional += temp[i].substr(ptr,temp[i].length())+"|";
```

```
        }
    }
    result.push_back(additional.substr(0,additional.length()-1));
    cout<<"----------------------------------------"<<endl;
    cout<<"Grammar After Left factoring"<<endl;
    for(auto i = result.begin() ; i!= result.end() ; i++){
        string s = *i;
        cout<<s<<endl;
    }
}

/*
    test cases :
    1.  A->xab|xc
    2.  A->abc|abcde
    3.  A->abcd| abed
    4.  A->ab|abc|abcd|axy
*/
```

Output:

```
(base) PS D:\DLP_lab\Practical_8> g++ .\left_factoring.cpp
(base) PS D:\DLP_lab\Practical_8> .\a.exe
Enter production rule:
[use CAPITAL for non-terminal and small case for terminal and ~ for NULL]
Exmaple: S->aB without space
A->ab|abc|abcd|axy
----------------------------------------
Grammar After Left factoring
A->aA'
A'->b|bc|bcd|xy
```

Conclusion: From this practical I have learnt about how to perform left factoring for given grammar.