LAB6: left factoring problem removal

IT057-Kathiriya Darshak

- Write a program to implement left factoring removal algorithm.

Input of program- Grammar with possibly the problem.

Output of program- Equivalent grammar without the problem.

Left factoring is a process by which the grammar with common prefixes is transformed

to make it useful for Top down parsers.

### **How?**

In left factoring,

* We make one production for each common prefixes.
* The common prefix may be a terminal or a non-terminal or a combination of both.
* Rest of the derivation is added by new productions.

The grammar obtained after the process of left factoring is called as **Left Factored Grammar**.

Code:

#include <bits/stdc++.h>

#include <iostream>

#include <vector>

#include <string>

using namespace std;

#define ll long long int

#define f(i,l,n) for(int i=l;i<n;i++)

#define E "\n"

#define fast ios::sync\_with\_stdio(0); cin.tie(0); cout.tie(0)

#define F first

#define S second

#define all(c) (c).begin(),(c).end()

#define sz(v) (int)(v).size()

#define vi(typ) vector<typ>

#define Ee cout<<"\n";

int j=1;

string decode(vector<string> S)

{

ll n=S.size();

vector<ll> indx(n+1,0);

vector<string> Print;

string Fur="";

f(o,1,n)

{

if(indx[o]==1 || S[o]=="\u03B5" || S[o].size()<=0) continue;

vector<string> L;

char c=S[o][0];

string F="",cur=S[o];

F=S[0];

f(times,0,j) F+='`';

L.push\_back(F);

F=c+F;

cur.erase(cur.begin());

if(cur.size()==0) cur="\u03B5";

L.push\_back(cur);

ll cnt=0;

f(i,1,n)

{

if(c==S[i][0] and o!=i)

{

string nn=S[i];

nn.erase(nn.begin());

L.push\_back(nn);

indx[i]=1;

cnt++;

}

}

if(cnt>0 and cnt<n-2)

{

indx[o]=1;

j++;

Print.push\_back(F);

Fur+=decode(L);

}

}

string ans=S[0];

ans+="-->";

bool yes=0;

for(auto x:Print) ans+=x+"|";

f(i,1,n) if(indx[i]==0) ans+=S[i]+"|";

ans.erase(ans.begin()+ans.size()-1);

ans+="\n";

ans+=Fur;

return ans;

}

ll solve()

{

vector<string> S;

string temp;

cin >> temp;

S.push\_back(temp);

ll n=0;

cin >> n;

f(i,0,n)

{

cin >> temp;

S.push\_back(temp);

}

string ans=decode(S);

cout<<ans<<E;

return 0;

}

int main()

{

printf("First line enter parent terminal \nSecond line enter number of production \nEnter production one by one\n");

ll t = 1;

fast;

while(t--)

{

int l=solve();

}

return 0;

}

Output

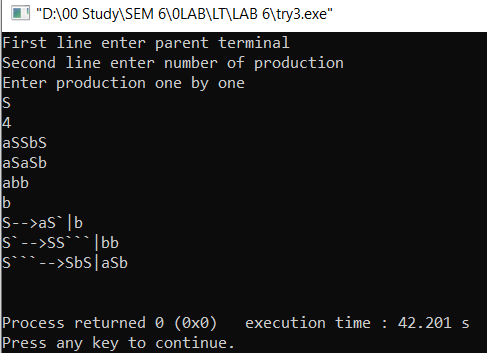
Input1: S --> bSSaaS|bSSaSb|bSb|a

Output:

S-->aS`|b

S`-->SS```|bb

S```-->SbS|aSb



Input2: S-->c|cL|L|cL+L

Output:

S-->cS`|L

S`-->LS```|#

S```-->#|+L

