Akshat Agarwal  
RA1911003010646

**Compiler Design**

**Lab – 6**

**ELIMINATION OF AMBIGUITY, LEFT RECURSION & Left factoring in the given grammar**

**b) Elimination of Left Recursion  
A -> A c | A a d | b d | ϵ**

**Code :**

#include<iostream>

#include<string>

using namespace std;

int main()

{ string ip,op1,op2,temp;

int sizes[10] = {};

char c;

int n,j,l;

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

cin>>c;

ip.push\_back(c);

op1 += ip + "\'->";

ip += "->";

op2+=ip;

cout<<"Enter the number of productions : ";

cin>>n;

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

{ cout<<"Enter Production "<<i+1<<" : ";

cin>>temp;

sizes[i] = temp.size();

ip+=temp;

if(i!=n-1)

ip += "|";

}

cout<<"Production Rule : "<<ip<<endl;

for(int i=0,k=3;i<n;i++)

{

if(ip[0] == ip[k])

{

cout<<"Production "<<i+1<<" has left recursion."<<endl;

if(ip[k] != '#')

{

for(l=k+1;l<k+sizes[i];l++)

op1.push\_back(ip[l]);

k=l+1;

op1.push\_back(ip[0]);

op1 += "\'|";

}

}

else

{

cout<<"Production "<<i+1<<" does not have left recursion."<<endl;

if(ip[k] != '#')

{

for(j=k;j<k+sizes[i];j++)

op2.push\_back(ip[j]);

k=j+1;

op2.push\_back(ip[0]);

op2 += "\'|";

}

else

{

op2.push\_back(ip[0]);

op2 += "\'";

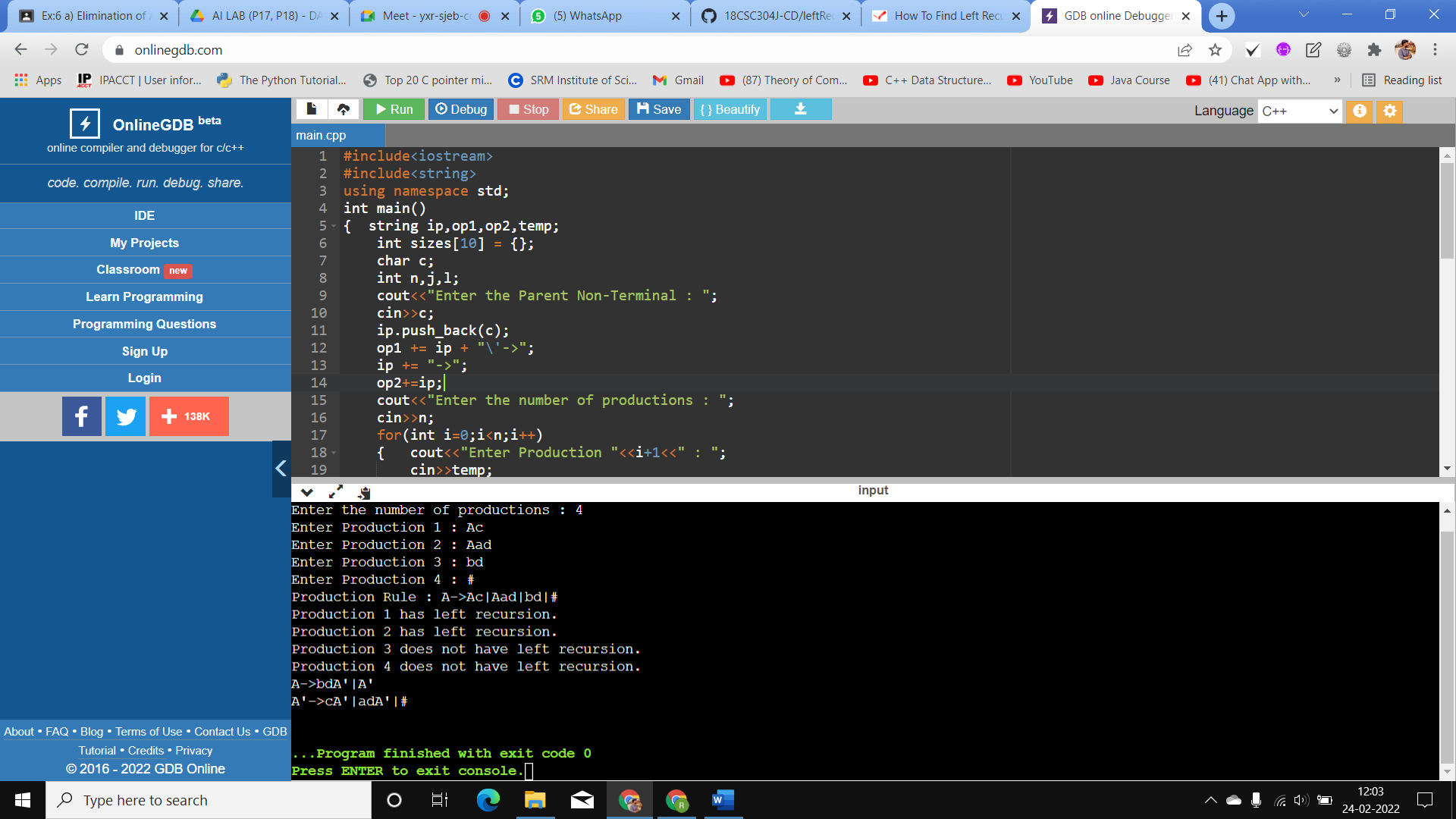
}}}

op1 += "#";

cout<<op2<<endl;

cout<<op1<<endl;

return 0;}

**OUTPUT :**

***c) Elimination of Left factoring  
i/p : M--> i | iM |(M)| iM+M***

**Code:**

#include <iostream>

#include <string>

using namespace std;

int main()

{

int n,j,l,i,m;

int len[10] = {};

string a, b1, b2, flag;

char c;

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

cin >> c;

a.push\_back(c);

b1 += a + "\'->";

b2 += a + "\'\'->";;

a += "->";

cout << "Enter total number of productions : ";

cin >> n;

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

{

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

cin >> flag;

len[i] = flag.size();

a += flag;

if (i != n - 1)

{

a += "|";

}

}

cout << "The Production Rule is : " << a << endl;

char x = a[3];

for (i = 0, m = 3; i < n; i++)

{

if (x != a[m])

{

while (a[m++] != '|');

}

else

{

if (a[m + 1] != '|')

{

b1 += "|" + a.substr(m + 1, len[i] - 1);

a.erase(m - 1, len[i] + 1);

}

else

{

b1 += "#";

a.insert(m + 1, 1, a[0]);

a.insert(m + 2, 1, '\'');

m += 4;

}

}

}

char y = b1[6];

for (i = 0, m = 6; i < n - 1; i++)

{

if (y == b1[m])

{

if (b1[m + 1] != '|')

{

flag.clear();

for (int s = m + 1; s < b1.length(); s++)

{

flag.push\_back(b1[s]);

}

b2 += "|" + flag;

b1.erase(m - 1, flag.length() + 2);

}

else

{

b1.insert(m + 1, 1, b1[0]);

b1.insert(m + 2, 2, '\'');

b2 += "#";

m += 5;

}

}

}

b2.erase(b2.size() - 1);

cout << "After Left Factoring : " << endl;

cout << a << endl;

cout << b1 << endl;

cout << b2 << endl;

return 0;

}

**Output :**

