17-Feb-22

**Experiment 4 - Left Recursion and Left Factoring**

Dhawal Patil

RA1911003010575

CSE A2

**Experiment 4a**

Aim:

A program for Elimination of Left Recursion

Algorithm:

1. Start the program.
2. Initialize the arrays for taking input from the user.
3. Prompt the user to input the no. of non-terminals having left recursion and no. of productions for these non-terminals.
4. Prompt the user to input the production for non-terminals.
5. Eliminate left recursion using the following rules:-

A->Aα1| Aα2 | . . . . . |Aαm

A->β1| β2| . . . . .| βn

Then replace it by

A-> βi A’ i=1,2,3,…..m

A’-> αj A’ j=1,2,3,…..n

A’-> Ɛ

1. After eliminating the left recursion by applying these rules, display the productions without left recursion.
2. Stop.

Code:

#include <iostream>

#include <vector>

#include <string>

using namespace std;

int main()

{

int n;

cout<<"\nEnter number of non terminals: ";

cin>>n;

cout<<"\nEnter non terminals one by one: ";

int i;

vector<string> nonter(n);

vector<int> leftrecr(n,0);

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

cout<<"\nNon terminal "<<i+1<<" : ";

cin>>nonter[i];

}

vector<vector<string> > prod;

cout<<"\nEnter '^' for null";

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

cout<<"\nNumber of "<<nonter[i]<<" productions: ";

int k;

cin>>k;

int j;

cout<<"\nOne by one enter all "<<nonter[i]<<" productions";

vector<string> temp(k);

for(j=0;j<k;++j) {

cout<<"\nRHS of production "<<j+1<<": ";

string abc;

cin>>abc;

temp[j]=abc;

if(nonter[i].length()<=abc.length()&&nonter[i].compare(abc.substr(0,nonter[i].length()))==0)

leftrecr[i]=1;

}

prod.push\_back(temp);

}

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

cout<<leftrecr[i];

}

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

if(leftrecr[i]==0)

continue;

int j;

nonter.push\_back(nonter[i]+"'");

vector<string> temp;

for(j=0;j<prod[i].size();++j) {

if(nonter[i].length()<=prod[i][j].length()&&nonter[i].compare(prod[i][j].substr(0,nonter[i].length()))==0) {

string abc=prod[i][j].substr(nonter[i].length(),prod[i][j].length()-nonter[i].length())+nonter[i]+"'";

temp.push\_back(abc);

prod[i].erase(prod[i].begin()+j);

--j;

}

else {

prod[i][j]+=nonter[i]+"'";

}

}

temp.push\_back("^");

prod.push\_back(temp);

}

cout<<"\n\n";

cout<<"\nNew set of non-terminals: ";

for(i=0;i<nonter.size();++i)

cout<<nonter[i]<<" ";

cout<<"\n\nNew set of productions: ";

for(i=0;i<nonter.size();++i) {

int j;

for(j=0;j<prod[i].size();++j) {

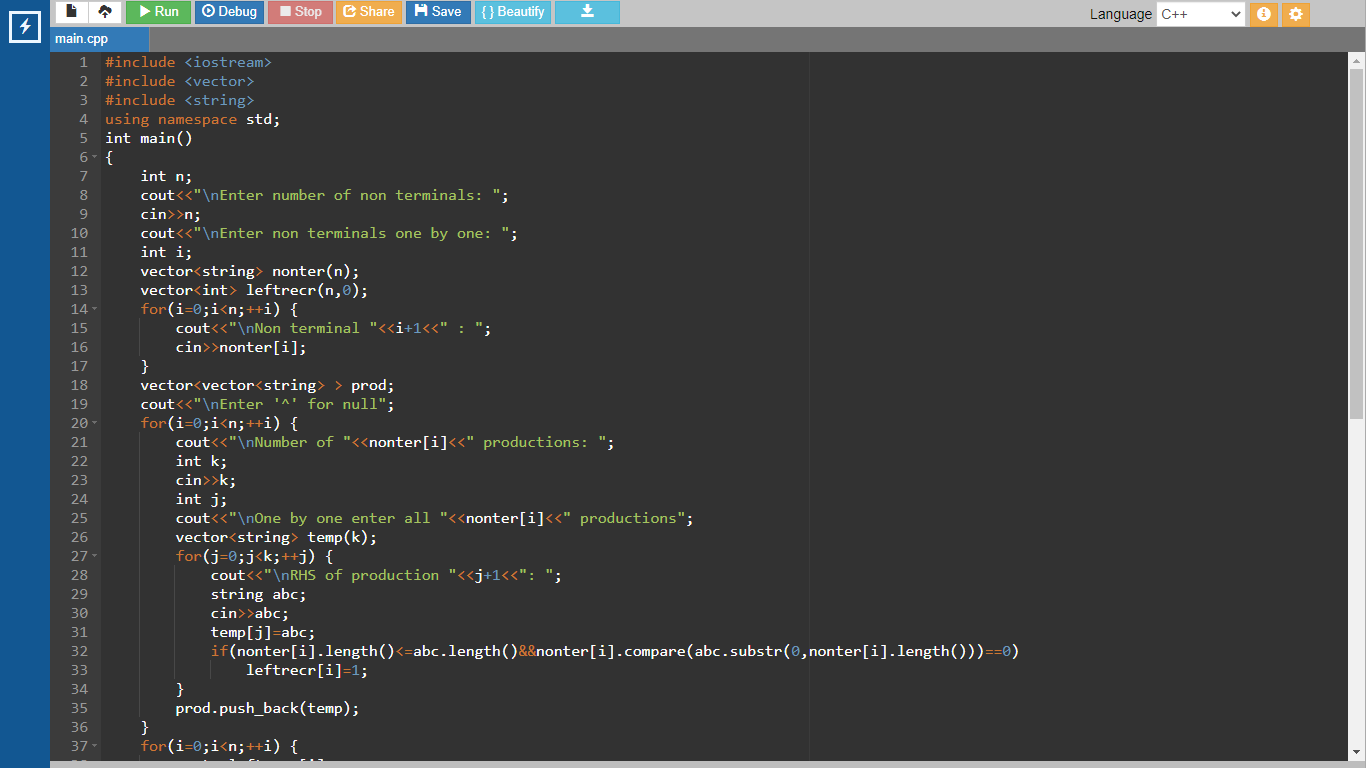
cout<<"\n"<<nonter[i]<<" -> "<<prod[i][j];

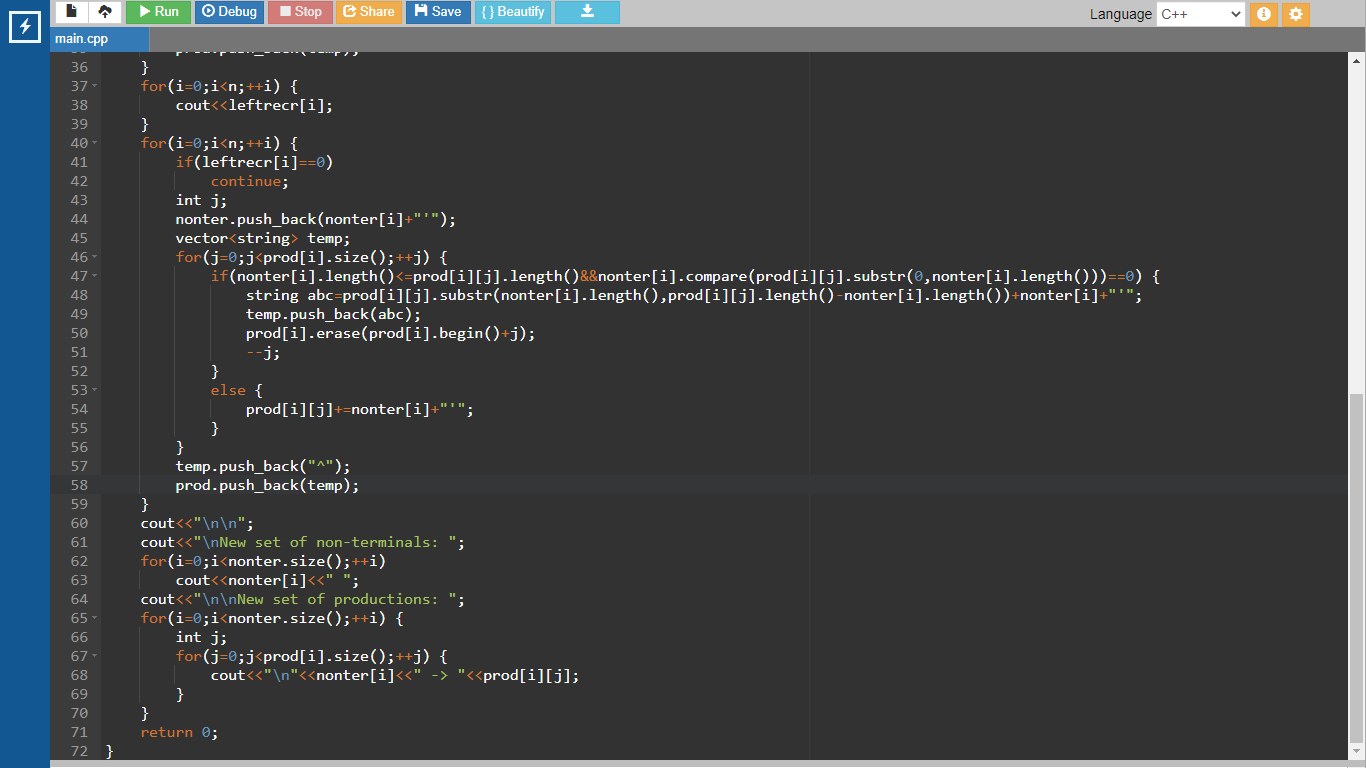
}

}

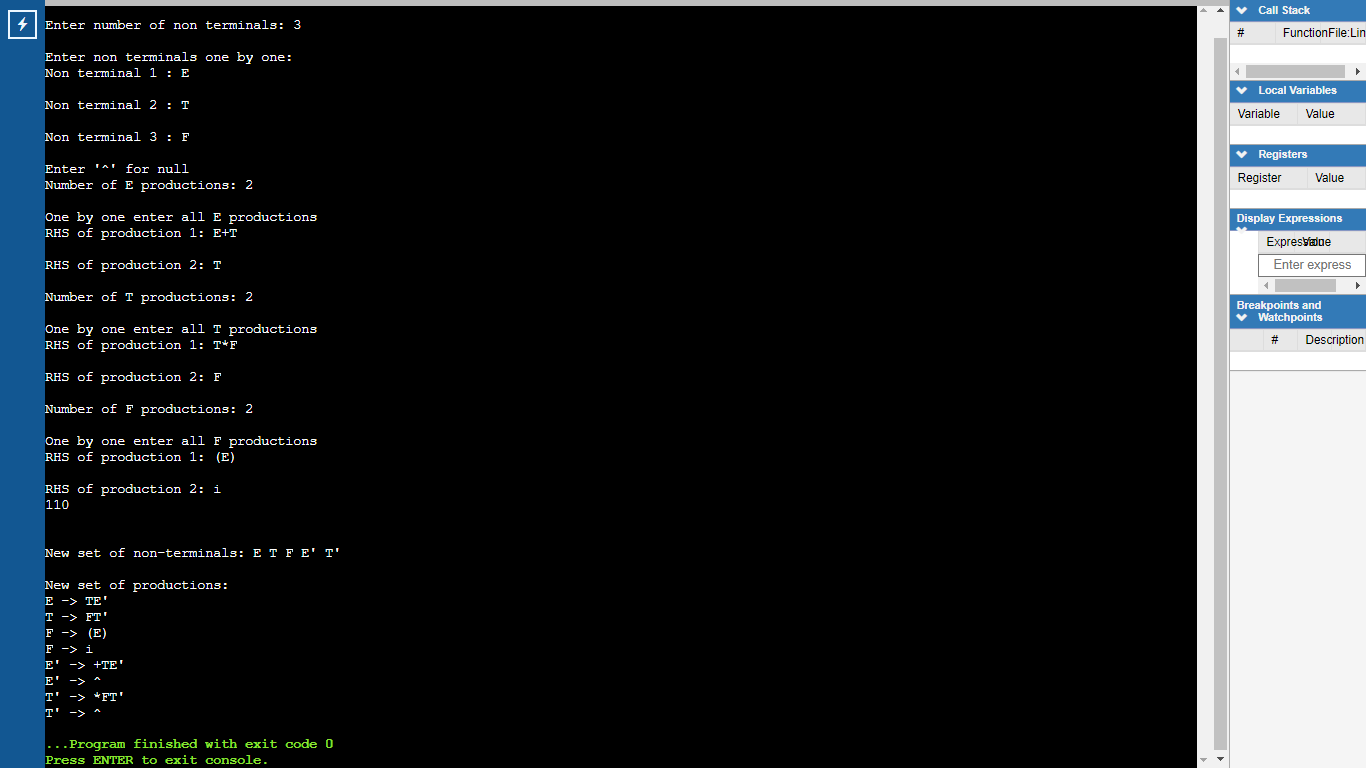
return 0;

}





Output:



Result:

A program for Elimination of Left Recursion was run successfully.

**Experiment 4b**

Aim:

A program for implementation Of Left Factoring

Algorithm:

1. Start
2. Ask the user to enter the set of productions
3. Check for common symbols in the given set of productions by comparing with:

A->aB1|aB2

1. If found, replace the particular productions with:

A->aA’

A’->B1 | B2|ɛ

1. Display the output
2. Exit

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;

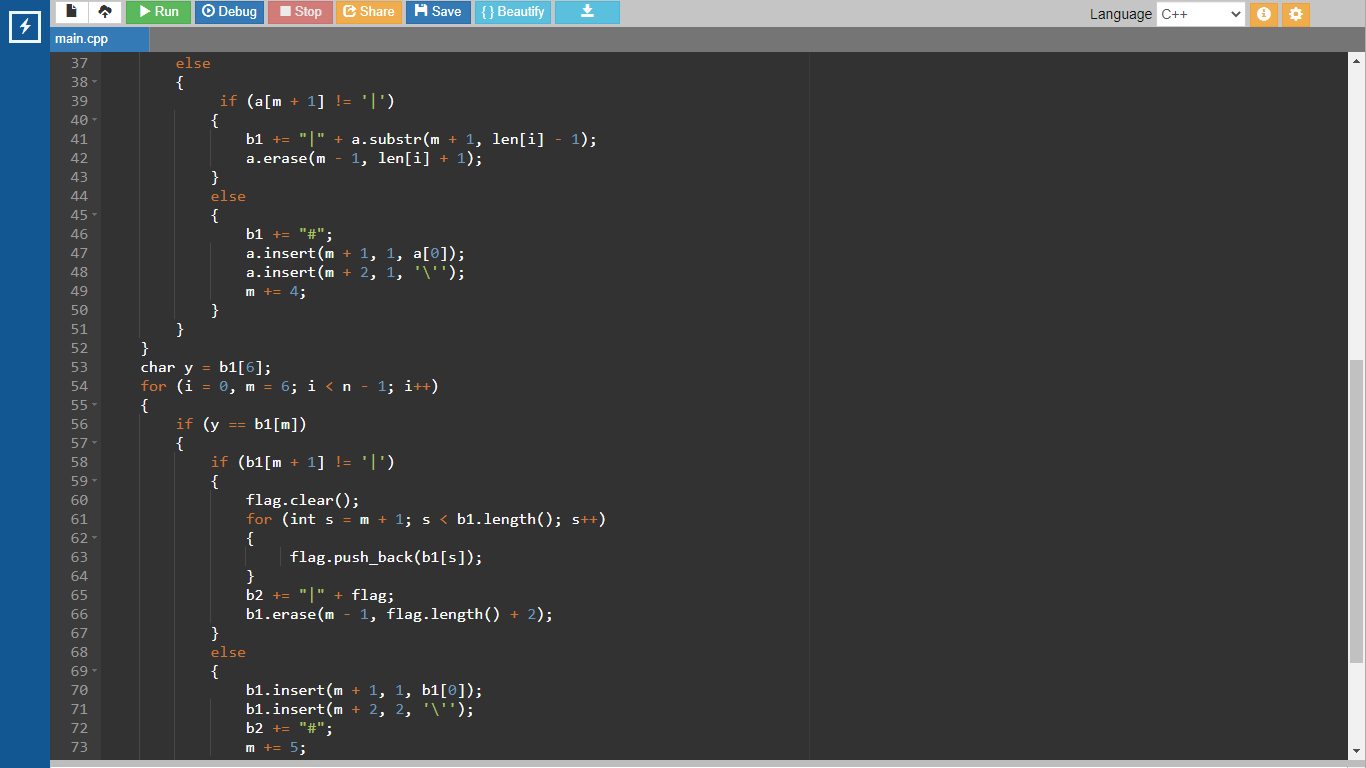
cout << b1 << endl;

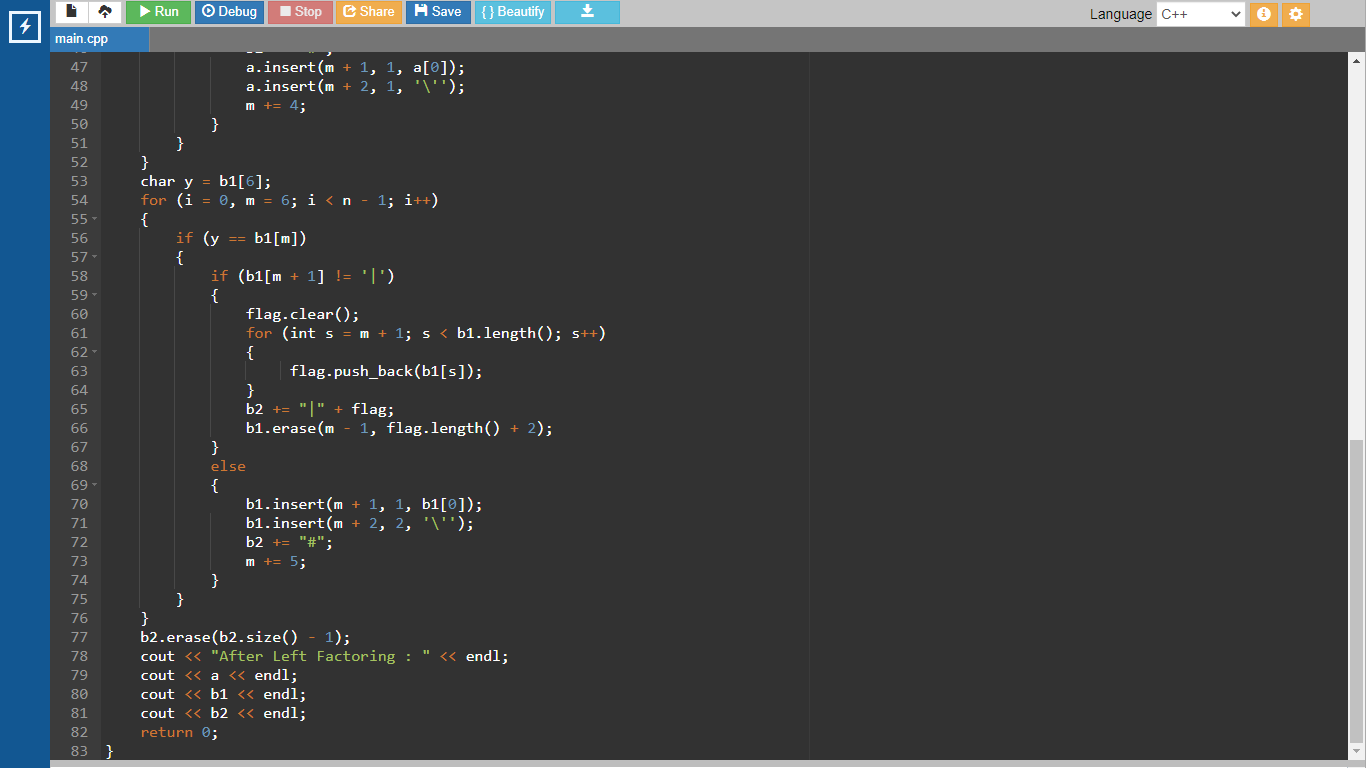
cout << b2 << endl;

return 0;

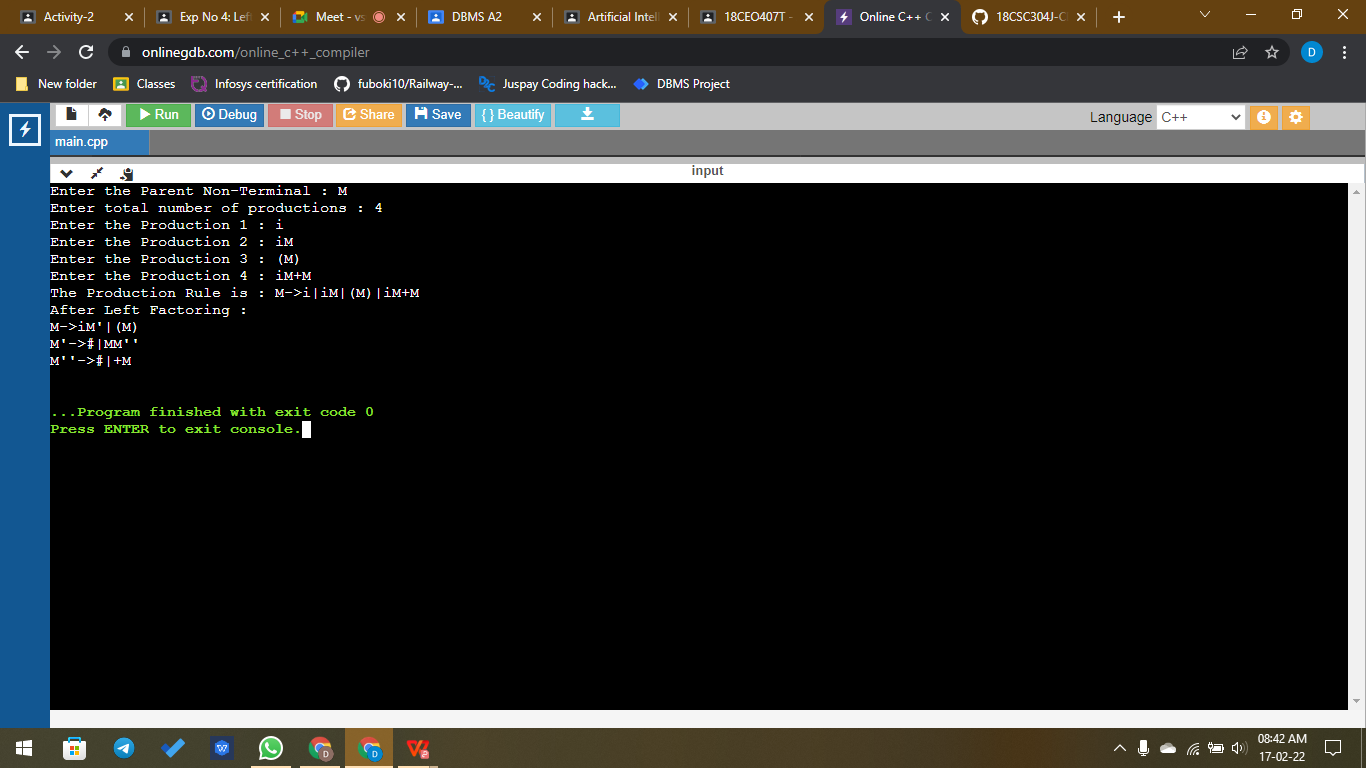
}







Output:



Result:

A program for implementation Of Left Factoring was compiled and run successfully