LEFT RECURSION

NAME: ATHRESH KUMAR LABDE
RA1911033010146
M1

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
6. After eliminating the left recursion by applying these rules, display the productions without left recursion.
7. Stop.

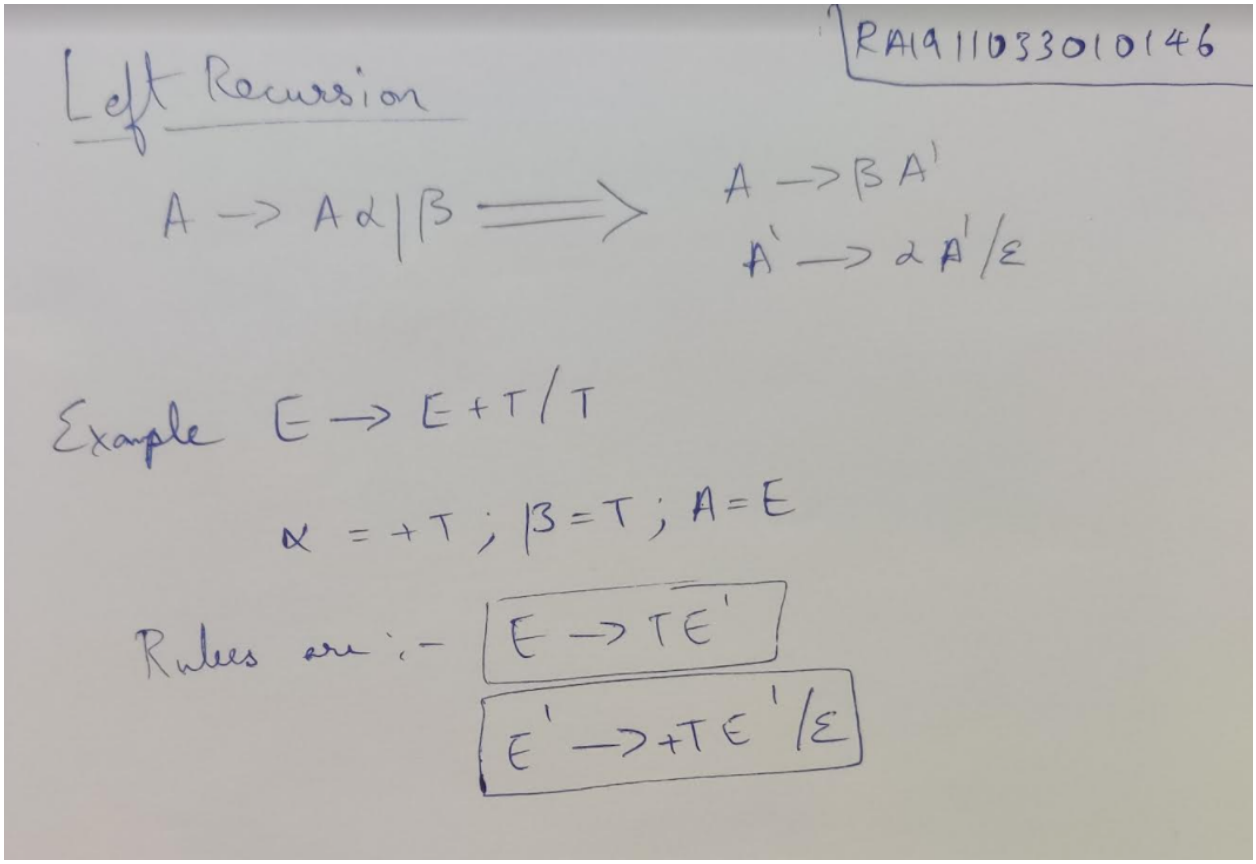
CODE:

```
#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].leng
		th())+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;
}
```

INPUT



OUTPUT

```
PROBLEMS 5   OUTPUT   DEBUG CONSOLE   TERMINAL

Enter number of non terminals: 1

Enter non terminals one by one:
Non terminal 1 : E

Enter '^' for null
Number of E productions: 2

One by one enter all E productions
RHS of production 1: E+T

RHS of production 2: T
1


New set of non-terminals: E E'

New set of productions:
E -> TE'
E' -> +TE'
E' -> ^
PS C:\Users\athre\Desktop\Athresh this sem\compiler design lab\5. recursion>
```

RESULT : A program for Elimination of Left Recursion was run successfully.