AIM:

To implement left recursion elimination for a production.

ALGORITHM:

- Get the production from the user.
- Use split function to split the LHS and RHS of the production and store it in k.
- Further split the RHS of the production at '/'.
- Run a loop on the productions on the right side and find the production with left recursion by calling the left recursion function.
- In the left recursive function, get the production which isn't left recursive and add a new non terminal to that.
- Finally, define the new non terminal and display the new non terminal production.
- Also, display the production which appends the new non terminal to the production which isn't left recursive.

C++ 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;
}
```

IMPLEMENTATION

```
Enter number of non terminals: 1

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

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

One by one enter all S productions
RHS of production 1: S0S1S

RHS of production 2: 01
1


New set of non-terminals: S S'

New set of productions:
S -> 01S'
S' -> 0S1SS'
S' -> ^

...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT

Code was successfully implemented and the output was verified.