LEFT FACTORING

NAME : ATHRESH KUMAR LABDE
RA1911033010146
M1

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
4. If found, replace the particular productions with:
   A->aA’
   A’->B1 | B2|ε
5. Display the output
6. Exit

CODE:

```
#include <iostream>
#include <math.h>
#include <vector>
#include <string>
#include <stdlib.h>
using namespace std;
int main()
{
	cout<<"\nEnter number of productions: ";
	int p;
	cin>>p;
	vector<string> prodleft(p),prodright(p);
	cout<<"\nEnter productions one by one: ";
	int i;
	for(i=0;i<p;++i) {
		cout<<"\nLeft of production "<<i+1<<": ";
		cin>>prodleft[i];
		cout<<"\nRight of production "<<i+1<<": ";
		cin>>prodright[i];
	}
 int j;
 int e=1;
 for(i=0;i<p;++i) {
	for(j=i+1;j<p;++j) {
		if(prodleft[j]==prodleft[i]) {
			int k=0;
			string com="";
```

```
                        while(k<prodright[i].length()&&k<prodright[j].length()&&prodright[i][k]==pro
                dright[j][k]) {
                        com+=prodright[i][k];
                        ++k;
        }
        if(k==0)
                continue;
                char* buffer;
                string comleft=prodleft[i];
        if(k==prodright[i].length()) {
                 prodleft[i]+=string(itoa(e,buffer,10));
                 prodleft[j]+=string(itoa(e,buffer,10));
                 prodright[i]="^";
                 prodright[j]=prodright[j].substr(k,prodright[j].length()-k);
         }
else if(k==prodright[j].length()) {
         prodleft[i]+=string(itoa(e,buffer,10));
         prodleft[j]+=string(itoa(e,buffer,10));
         prodright[j]="^";
         prodright[i]=prodright[i].substr(k,prodright[i].length()-k);
}
else {
         prodleft[i]+=string(itoa(e,buffer,10));
         prodleft[j]+=string(itoa(e,buffer,10));
         prodright[j]=prodright[j].substr(k,prodright[j].length()-k);
         prodright[i]=prodright[i].substr(k,prodright[i].length()-k);
}
int l;
for(l=j+1;l<p;++l) {
        if(comleft==prodleft[l]&&com==prodright[l].substr(0,fmin(k,prodright[l].length()))) {
         prodleft[l]+=string(itoa(e,buffer,10));
         prodright[l]=prodright[l].substr(k,prodright[l].length()-k);
         }
}
         prodleft.push_back(comleft);
         prodright.push_back(com+prodleft[i]);
         ++p;
         ++e;
         }
    }
}
cout<<"\n\nNew productions";
for(i=0;i<p;++i) {
         cout<<"\n"<<prodleft[i]<<"->"<<prodright[i];
```

```
 }
 return 0;
}
```

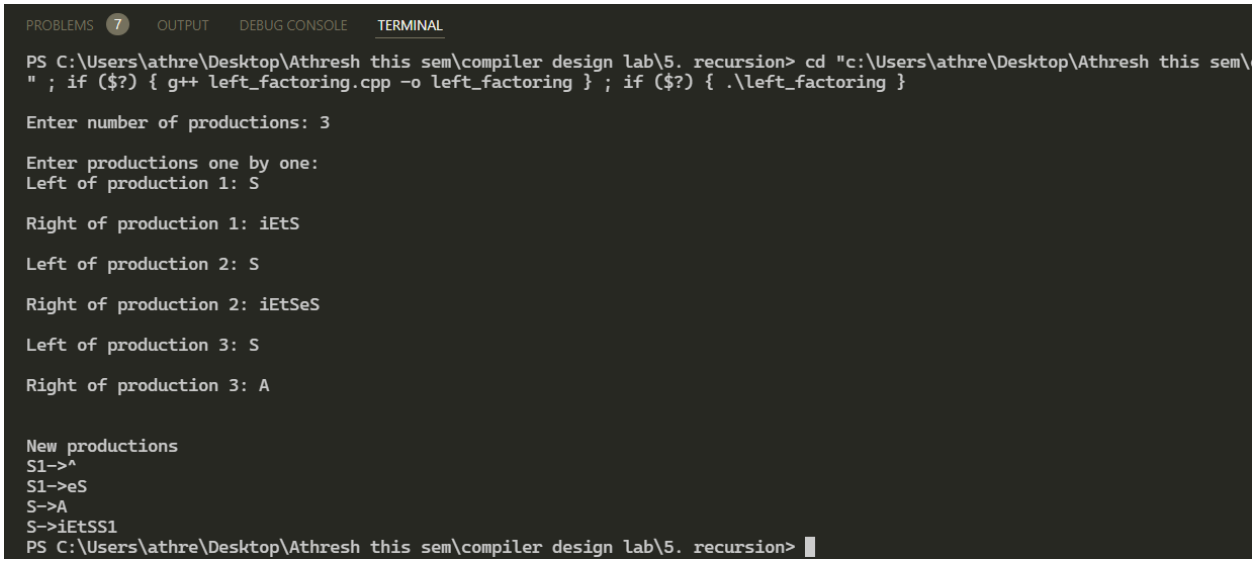
INPUT :

S → iEtS / iEtSeS / a

OUTPUT :

S → iEtSS’ / a

S’ → eS / ∈



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