**Compiler Design(18CSC304J)**

**Experiment 6**

**ELIMINATION OF LEFT RECURSION**

Harsh Goel

RA1811003010185

**Aim:** To study and implement Elimination of Left Recursion.

**Language: C++**

**Theory:**

A production of grammar is said to have left recursion if the leftmost variable of its RHS is same as variable of its LHS.

A grammar containing a production having left recursion is called as Left Recursive Grammar.

**Procedure:**

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:-
6. A->Aα1| Aα2 | . . . . . |Aαm
7. A->β1| β2| . . . . .| βn
8. Then replace it by
9. A-> βi A’ i=1,2,3,…..m
10. A’-> αj A’ j=1,2,3,…..n
11. A’-> Ɛ
12. After eliminating the left recursion by applying these rules, display the productions without left recursion.
13. Stop

**Code Snippet:**

#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 'esp' 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("esp");

        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) {

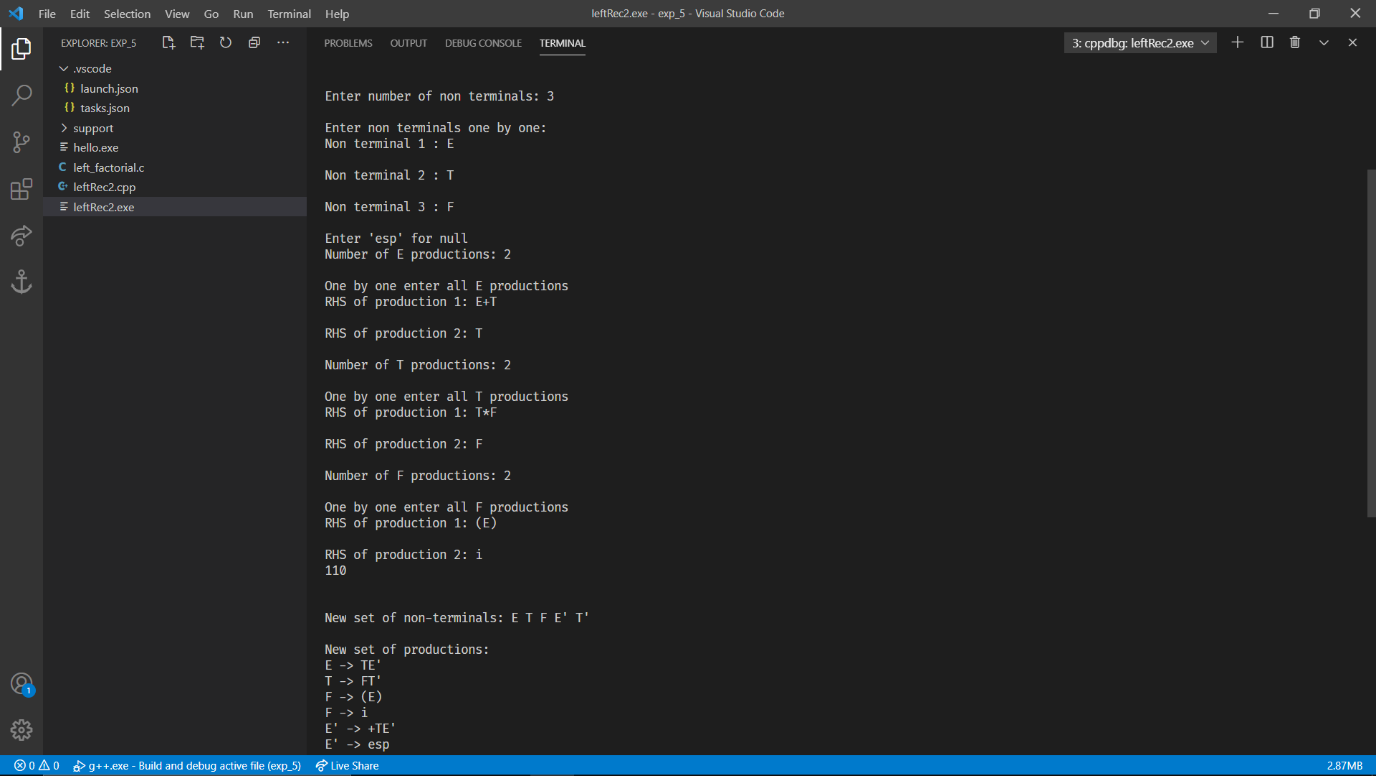
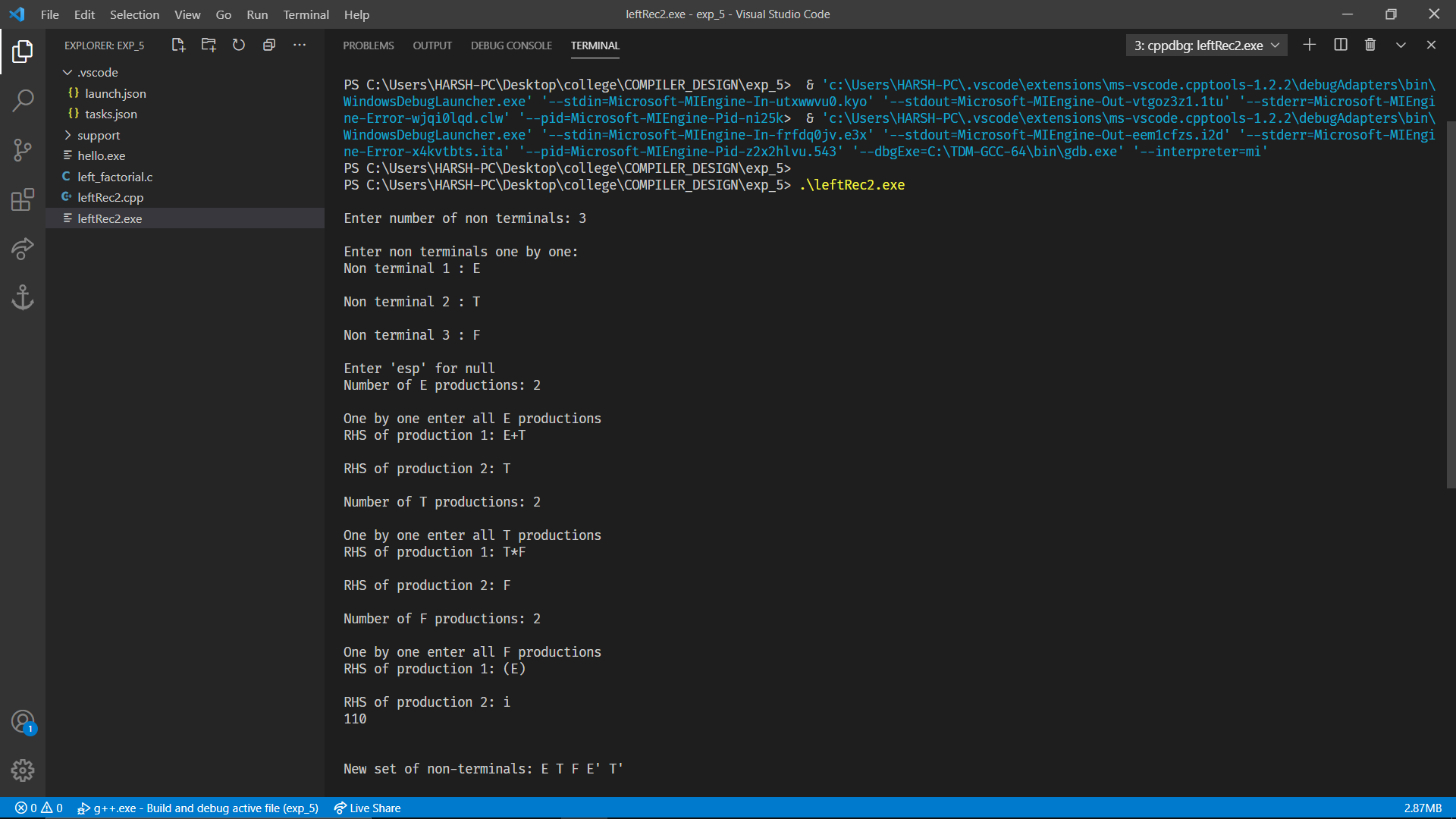
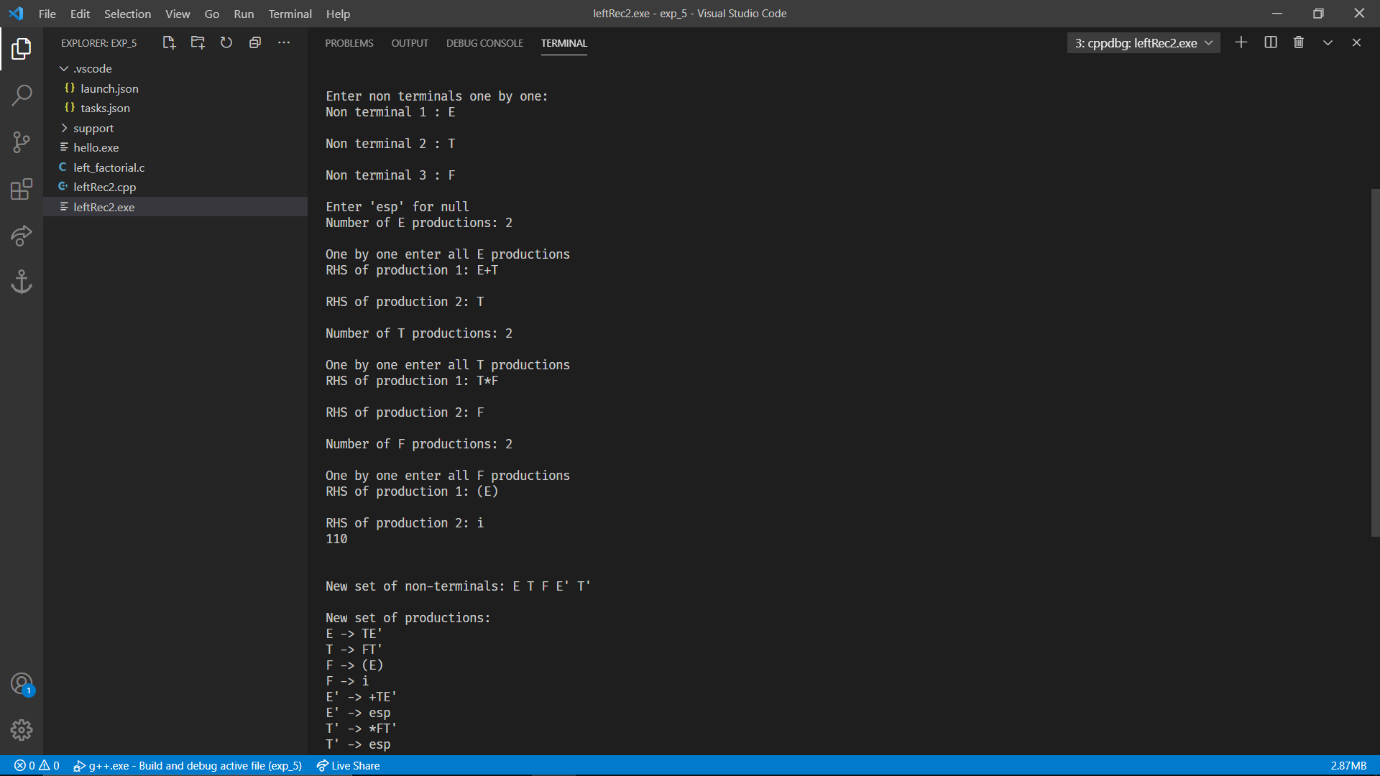
            cout<<"\n"<<nonter[i]<<" -> "<<prod[i][j];

        }

    }

    return 0;

}

**Output Screenshots:** 

**Result:**

The code was successfully implemented in C and output was recorded. Hence, A program for Elimination of Left Recursion was run successfully.