RA2011031010046

EXPERIMENT : 4a **ELIMINATION OF AMBIGUITY USING LEFT RECURSION**

DATE : 09/02/23

AIM : To write a program in C/C++ to eliminate ambiguity using left recursion.

PROCEDURE:

1. Start the program.
2. Initialize the arrays for taking input from the user.
3. Prompt the user to input the number of non-terminals having left recursion and number of production 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
   B -> β1 | β2 | ….βn
   Then replace it by:
   A -> βiA' i = 1,2,3,..m
   A' -> αjA' j = 1,2,3…n
   A' -> ε
6. After eliminating the left recursion by applying these rules, display the productions.

PROGRAM:

```
#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;
    }
```

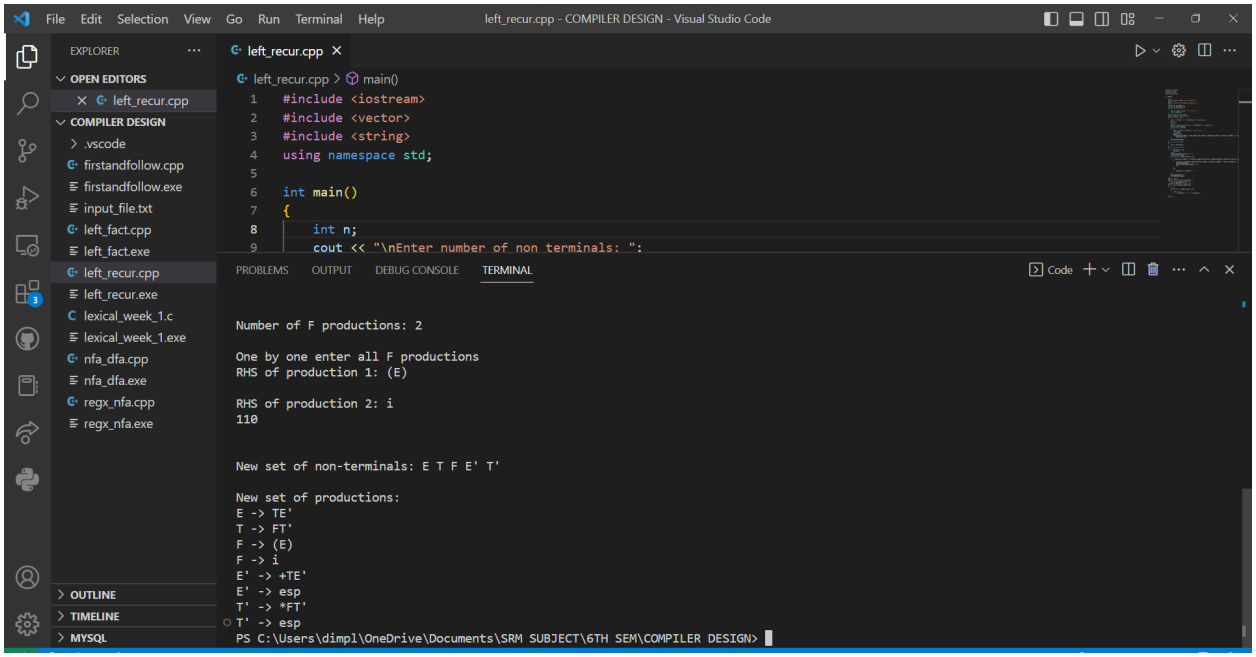
INPUT:

```
Enter number of non terminals: 3

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

Non terminal 2 : T

Non terminal 3 : F

Enter 'esp' 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

Number of T productions: 2

One by one enter all T productions
RHS of production 1: T*F

RHS of production 2: F

Number of F productions: 2
```

One by one enter all F productions
RHS of production 1: (E)

RHS of production 2: i
110

OUTPUT:



RESULT:

Thus, we have successfully implemented the concept of ambiguity elimination using left recursion.