Name: Atandrit Chatterjee

Reg. No.: RA2011031010042

Experiment-4 ELIMINATION OF AMBIGUITY

Aim:

To write a program implementing elimination of ambiguity using Left Recursion and Left Factoring.

Procedure:

a. Elimination of Left Recursion:

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.

b. Implementation of Left Factoring:

```
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
```

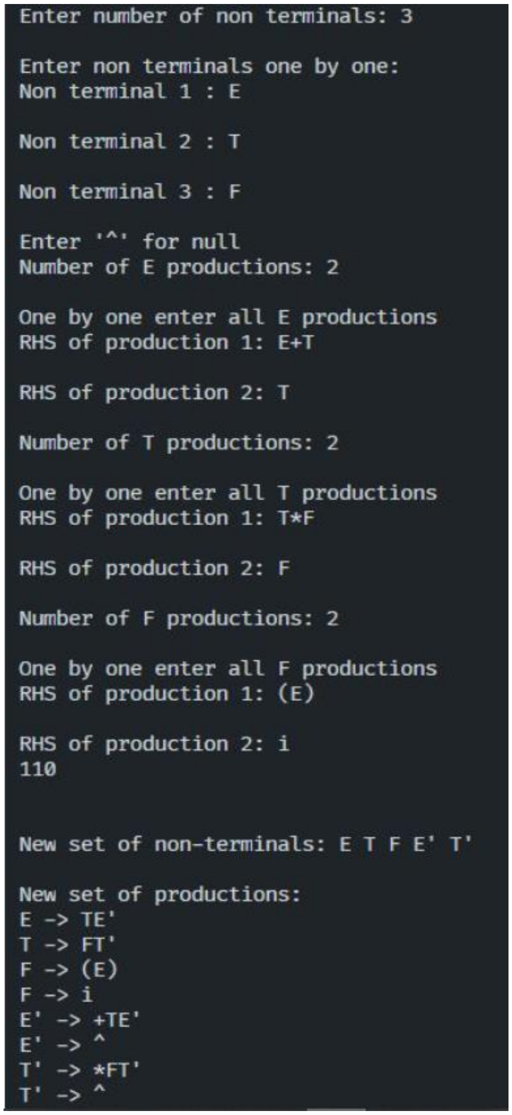
Code:

a. Elimination of Left Recursion:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    system("cls");
    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];
        }
    }
    system("pause");
    return 0;
}
```

Output:



```
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 '^' 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


New set of non-terminals: E T F E' T'

New set of productions:
E -> TE'
T -> FT'
F -> (E)
F -> i
E' -> +TE'
E' -> ^
T' -> *FT'
T' -> ^
```

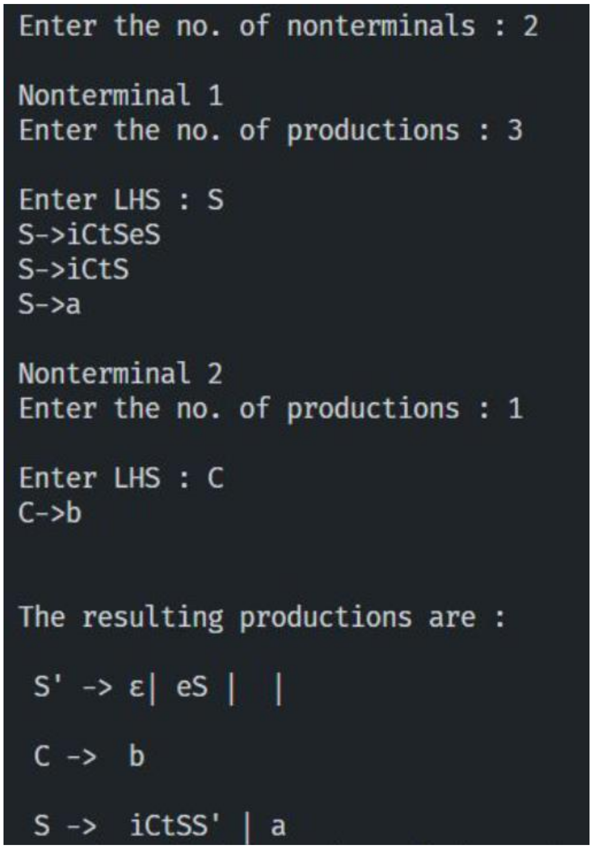
b. Implementation of Left Factoring:

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main()
{
    system("cls");
    char ch, lhs[20][20], rhs[20][20][20], temp[20], temp1[20];
    int n, n1, count[20], x, y, i, j, k, c[20];
    printf("\nEnter the no. of nonterminals : ");
    scanf("%d", &n);
    n1 = n;
    for (i = 0; i < n; i++)
    {
        printf("\nNonterminal %d \nEnter the no. of productions : ", i + 1);
        scanf("%d", &c[i]);
        printf("\nEnter LHS : ");
        scanf("%s", lhs[i]);
        for (j = 0; j < c[i]; j++)
        {
            printf("%s->", lhs[i]);
            scanf("%s", rhs[i][j]);
        }
    }
    for (i = 0; i < n; i++)
    {
        count[i] = 1;
        while (memcmp(rhs[i][0], rhs[i][1], count[i]) == 0)
            count[i]++;
```

```
}
for (i = 0; i < n; i++)
{
    count[i]--;
    if (count[i] > 0)
    {
        strcpy(lhs[n1], lhs[i]);
        strcat(lhs[i], "'");
        for (k = 0; k < count[i]; k++)
            temp1[k] = rhs[i][0][k];
        temp1[k++] = '\0';
        for (j = 0; j < c[i]; j++)
        {
            for (k = count[i], x = 0; k < strlen(rhs[i][j]); x++, k++)
                temp[x] = rhs[i][j][k];
            temp[x++] = '\0';
            if (strlen(rhs[i][j]) == 1)
                strcpy(rhs[n1][1], rhs[i][j]);
            strcpy(rhs[i][j], temp);
        }
        c[n1] = 2;
        strcpy(rhs[n1][0], temp1);
        strcat(rhs[n1][0], lhs[n1]);
        strcat(rhs[n1][0], "'");
        n1++;
    }
}
printf("\n\nThe resulting productions are : \n");
for (i = 0; i < n1; i++)
{
    if (i == 0)
        printf("\n %s -> %c|", lhs[i], (char)238);
    else
```

```
            printf("\n %s -> ", lhs[i]);
        for (j = 0; j < c[i]; j++)
        {
            printf(" %s ", rhs[i][j]);
            if ((j + 1) != c[i])
                printf("|");
        }
        printf("\b\b\b\n");
    }
    return 0;
}
```

Output:



```
Enter the no. of nonterminals : 2

Nonterminal 1
Enter the no. of productions : 3

Enter LHS : S
S->iCtSeS
S->iCtS
S->a

Nonterminal 2
Enter the no. of productions : 1

Enter LHS : C
C->b


The resulting productions are :

 S' -> ε| eS |  |

 C ->  b

 S ->  iCtSS' | a
```

Result:

Elimination of ambiguity using Left Recursion and Left Factoring has been done successfully.