Madhurya Mozumder

RA1911028010036

CSE-CC I2

**Experiment 4**

**Elimination of Left Recursion**

**Aim:** Write a program in your preferred language to eliminate the left recursion in the given production rules

# Algorithm:

1. Check if the given grammar contains left recursion, if present then separates the production and start working on it.
2. Introduce a new nonterminal and write it at the last of every terminal.
3. Write newly produced nonterminal in LHS and in RHS it can either produce or it can produce new production in which the terminals or non-terminals which followed the previous LHS will be replaced by new nonterminal at last.

# Program:

# #include <iostream>

# #include <string>

# using namespace std;

# int main()

# {

# int n, j, l, i, k;

# int length[10] = {};

# string d, a, b, flag;

# char c;

# cout<<"Enter Parent Non-Terminal: ";

# cin >> c;

# d.push\_back(c);

# a += d + "\'->";

# d += "->";

# b += d;

# cout<<"Enter productions: ";

# cin >> n;

# for (int i = 0; i < n; i++)

# {

# cout<<"Enter Production ";

# cout<<i + 1<<" :";

# cin >> flag;

# length[i] = flag.size();

# d += flag;

# if (i != n - 1)

# {

# d += "|";

# }

# }

# cout<<"The Production Rule is: ";

# cout<<d<<endl;

# for (i = 0, k = 3; i < n; i++)

# {

# if (d[0] != d[k])

# {

# cout<<"Production: "<< i + 1;

# cout<<" does not have left recursion.";

# cout<<endl;

# if (d[k] == '#')

# {

# b.push\_back(d[0]);

# b += "\'";

# }

# else

# {

# for (j = k; j < k + length[i]; j++)

# {

# b.push\_back(d[j]);

# }

# k = j + 1;

# b.push\_back(d[0]);

# b += "\'|";

# }

# }

# else

# {

# cout<<"Production: "<< i + 1 ;

# cout<< " has left recursion";

# cout<< endl;

# if (d[k] != '#')

# {

# for (l = k + 1; l < k + length[i]; l++)

# {

# a.push\_back(d[l]);

# }

# k = l + 1;

# a.push\_back(d[0]);

# a += "\'|";

# }

# }

# }

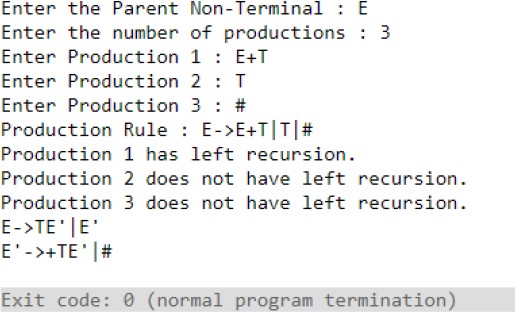
# a += "#";

# cout << b << endl;

# cout << a << endl;

# return 0;

# }Input /Output:



**Result:** A program was written down in C for the Implementation for the elimination of the left recursion.

**Aim:** Write a program in your preferred language to perform left factoring in the given production rules.

# Algorithm:

1. For all A ∈ non-terminal, find the longest prefix a that occurs in two or more right-hand sides of A.
2. If a ¹ ∈ then replace all of the A productions, A → a bI | a b2 | - - - | a bn | r with
3. A → a A | r A → bI | b2| - - - | bn | ∈ Where, A is a new element of non-terminal.
4. Repeat until no common prefixes remain.

# Program:

# #include <iostream>

# #include <string>

# using namespace std;

# int main()

# {

# int n,j,l,i,m;

# int len[10] = {};

# string a, b1, b2, flag;

# char c;

# cout << "Enter the Parent Non-Terminal : ";

# cin >> c;

# a.push\_back(c);

# b1 += a + "\'->";

# b2 += a + "\'\'->";;

# a += "->";

# cout << "Enter total number of productions : ";

# cin >> n;

# for (i = 0; i < n; i++)

# {

# cout << "Enter the Production " << i + 1 << " : ";

# cin >> flag;

# len[i] = flag.size();

# a += flag;

# if (i != n - 1)

# {

# a += "|";

# }

# }

# cout << "The Production Rule is : " << a << endl;

# char x = a[3];

# for (i = 0, m = 3; i < n; i++)

# {

# if (x != a[m])

# {

# while (a[m++] != '|');

# }

# else

# {

# if (a[m + 1] != '|')

# {

# b1 += "|" + a.substr(m + 1, len[i] - 1);

# a.erase(m - 1, len[i] + 1);

# }

# else

# {

# b1 += "#";

# a.insert(m + 1, 1, a[0]);

# a.insert(m + 2, 1, '\'');

# m += 4;

# }

# }

# }

# char y = b1[6];

# for (i = 0, m = 6; i < n - 1; i++)

# {

# if (y == b1[m])

# {

# if (b1[m + 1] != '|')

# {

# flag.clear();

# for (int s = m + 1; s < b1.length(); s++)

# {

# flag.push\_back(b1[s]);

# }

# b2 += "|" + flag;

# b1.erase(m - 1, flag.length() + 2);

# }

# else

# {

# b1.insert(m + 1, 1, b1[0]);

# b1.insert(m + 2, 2, '\'');

# b2 += "#";

# m += 5;

# }

# }

# }

# b2.erase(b2.size() - 1);

# cout << "After Left Factoring : " << endl;

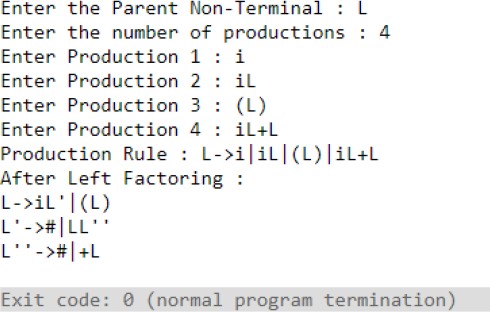
# cout << a << endl;

# cout << b1 << endl;

# cout << b2 << endl;

# return 0;

# }Input /Output:



**Result:** A program was written down in C for performing left factoring on the given rules.