**Lab experiment 6th question**

**6.** **Implement a C program to eliminate left recursion****.**

**Aim:**

To develop a **C program** that eliminates **left recursion** from a given grammar. **Left recursion** occurs when a non-terminal symbol on the left side of a production rule also appears at the beginning of its right side.

**Procedure:**

1. **Understand Left Recursion:**
   * A grammar has **left recursion** if a rule is of the form:

less

CopyEdit

A → Aα | β

where A is a non-terminal, α is a sequence of terminals/non-terminals, and β does **not** start with A.

* + Left recursion causes infinite loops in **top-down parsing** (like Recursive Descent Parsing).

1. **Convert Left Recursion to Right Recursion:**
   * Convert the recursive rule into a right-recursive form:

vbnet

CopyEdit

A → βA'

A' → αA' | ε

* + Here, A' is a new non-terminal, and ε represents an **empty string (null production)**.

1. **Write a C Program:**
   * Read a grammar rule.
   * Identify if it has left recursion.
   * Convert it into a right-recursive form.
   * Display the transformed grammar.

**C Program Implementation**

This program eliminates **immediate left recursion** for a single non-terminal.

c

CopyEdit

#include <stdio.h>

#include <string.h>

void eliminateLeftRecursion(char nonTerminal, char alpha[], char beta[]) {

char newNonTerminal = nonTerminal + '1'; // Create a new non-terminal (e.g., A → A')

printf("After eliminating left recursion:\n");

printf("%c → %s%c'\n", nonTerminal, beta, newNonTerminal);

printf("%c' → %s%c' | ε\n", newNonTerminal, alpha, newNonTerminal);

}

int main() {

char nonTerminal, alpha[20], beta[20];

// Input: Grammar rule in form A -> Aα | β

printf("Enter the non-terminal (A-Z): ");

scanf(" %c", &nonTerminal);

printf("Enter left recursive part (α): ");

scanf("%s", alpha);

printf("Enter non-recursive part (β): ");

scanf("%s", beta);

// Display original grammar

printf("\nOriginal Grammar:\n");

printf("%c → %c%s | %s\n", nonTerminal, nonTerminal, alpha, beta);

// Call function to eliminate left recursion

eliminateLeftRecursion(nonTerminal, alpha, beta);

return 0;

}