Lab experiment 7th question

7. Implement a C program to eliminate left factoring

**Aim:**

To develop a **C program** that eliminates **left factoring** in a given grammar.

**Procedure:**

**1. Understand Left Factoring:**

* Left factoring occurs when multiple productions of a non-terminal have a **common prefix**:

less

CopyEdit

A → αβ1 | αβ2

Here, α is the common prefix, and β1, β2 are different suffixes.

* This can be rewritten as:

vbnet

CopyEdit

A → αA'

A' → β1 | β2

Where A' is a **new non-terminal** representing the choices after α.

**2. Steps to Remove Left Factoring:**

1. **Identify a common prefix** in multiple productions of a non-terminal.
2. **Extract the common prefix (α)** and introduce a **new non-terminal (A')**.
3. **Rewrite the grammar** using A' to handle the alternatives (β1, β2).

PROGRAM:

#include <stdio.h>

#include <string.h>

void eliminateLeftFactoring(char nonTerminal, char alpha[], char beta1[], char beta2[]) {

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

printf("\nAfter eliminating left factoring:\n");

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

printf("%c' → %s | %s\n", newNonTerminal, beta1, beta2);

}

int main() {

char nonTerminal, alpha[20], beta1[20], beta2[20];

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

scanf(" %c", &nonTerminal);

printf("Enter common prefix (α): ");

scanf("%s", alpha);

printf("Enter first alternative (β1): ");

scanf("%s", beta1);

printf("Enter second alternative (β2): ");

scanf("%s", beta2);

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

printf("%c → %s%s | %s%s\n", nonTerminal, alpha, beta1, alpha, beta2);

eliminateLeftFactoring(nonTerminal, alpha, beta1, beta2);

return 0;

}

OUTPUT:

