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

**AIM:**

To implement a C program to eliminate left recursion in context-free grammar rules.

**Algorithm**

1. **Start**
2. Input the grammar rules in the form A → Aα | β, where A is a non-terminal symbol, α and β are strings of terminals/non-terminals.
3. Check if the grammar contains left recursion:
   * If a production starts with the same non-terminal as on the left side of the rule (A → Aα), it indicates left recursion.
4. If left recursion is detected:
   * Remove left recursion by rewriting the rules as:

vbnet

CopyEdit

A → βA'

A' → αA' | ε

Where A' is a new non-terminal symbol and ε represents an empty string.

1. Print the updated grammar without left recursion.
2. **End**

**CODE:**

#include <stdio.h>

#include <string.h>

#define MAX 100

void eliminateLeftRecursion(char nonTerminal, char \*production) {

char alpha[MAX], beta[MAX];

int i = 0, j = 0, k = 0;

if (production[0] == nonTerminal) {

i = 1;

while (production[i] != '|' && production[i] != '\0') {

alpha[j++] = production[i++];

}

alpha[j] = '\0';

i++;

while (production[i] != '\0') {

beta[k++] = production[i++];

}

beta[k] = '\0';

printf("Grammar after eliminating left recursion:\n");

printf("%c -> %s%c'\n", nonTerminal, beta, nonTerminal + 1);

printf("%c' -> %s%c' | ε\n", nonTerminal + 1, alpha, nonTerminal + 1);

} else {

printf("The production does not have left recursion.\n");

}

}

int main() {

char nonTerminal;

char production[MAX];

printf("Enter the non-terminal (e.g., A): ");

scanf(" %c", &nonTerminal);

printf("Enter the production (format: Aalpha|beta): ");

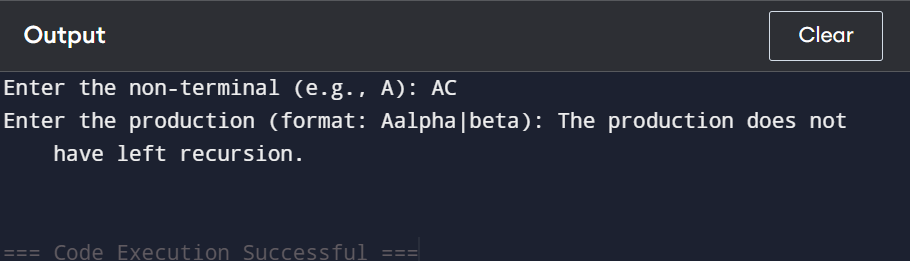
scanf(" %s", production);

eliminateLeftRecursion(nonTerminal, production);

return 0;

}

**OUTPUT:**

****