9) **Implement a C program to eliminate left recursion from a given CFG?**

#include <stdio.h>

#include <string.h>

#define MAX 100

void eliminateLeftRecursion(char grammar[][MAX], int n) {

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

char nonTerminal = grammar[i][0];

char alpha[MAX][MAX], beta[MAX][MAX];

int alphaCount = 0, betaCount = 0;

// Separate alpha and beta productions

char \*production = strtok(grammar[i], " ");

production = strtok(NULL, " ");

while (production != NULL) {

if (production[0] == nonTerminal) {

strcpy(alpha[alphaCount++], &production[1]);

} else {

strcpy(beta[betaCount++], production);

}

production = strtok(NULL, " ");

}

// If left recursion is present

if (alphaCount > 0) {

printf("%c -> ", nonTerminal);

for (int j = 0; j < betaCount; j++) {

printf("%s%c' ", beta[j], nonTerminal);

}

printf("\n");

printf("%c' -> ", nonTerminal);

for (int j = 0; j < alphaCount; j++) {

printf("%s%c' ", alpha[j], nonTerminal);

}

printf("ε\n");

} else {

printf("%s\n", grammar[i]);

}

}

}

int main() {

int n;

printf("Enter the number of productions: ");

scanf("%d", &n);

char grammar[MAX][MAX];

printf("Enter the productions (e.g., E E+T | T): \n");

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

scanf(" %[^\n]", grammar[i]);

}

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

eliminateLeftRecursion(grammar, n);

return 0;

}

