Implement a C program to eliminate left factoring from a given CFG.

**Code:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX\_RULES 10

#define MAX\_LENGTH 100

void eliminateLeftFactoring(char rules[MAX\_RULES][MAX\_LENGTH], int ruleCount) {

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

char \*production = strstr(rules[i], "->");

if (!production) continue;

\*production = '\0'; // Split LHS and RHS

char lhs[MAX\_LENGTH], rhs[MAX\_LENGTH];

strcpy(lhs, rules[i]);

strcpy(rhs, production + 2);

char \*tokens[MAX\_RULES];

int tokenCount = 0;

char \*token = strtok(rhs, "|");

while (token) {

tokens[tokenCount++] = token;

token = strtok(NULL, "|");

}

int prefixLen = strlen(tokens[0]);

for (int j = 1; j < tokenCount; j++) {

int k = 0;

while (k < prefixLen && tokens[0][k] == tokens[j][k]) k++;

prefixLen = k;

}

if (prefixLen == 0) { // No left factoring

printf("%s->%s\n", lhs, rhs);

continue;

}

char prefix[MAX\_LENGTH];

strncpy(prefix, tokens[0], prefixLen);

prefix[prefixLen] = '\0';

printf("%s->%s%c'\n", lhs, prefix, lhs[0]);

printf("%c'->", lhs[0]);

int first = 1;

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

if (strncmp(tokens[j], prefix, prefixLen) == 0) {

if (!first) printf("|");

printf("%s", tokens[j] + prefixLen);

first = 0;

}

}

printf("\n");

}

}

int main() {

char rules[MAX\_RULES][MAX\_LENGTH] = {

"A->ab|ac|ad",

"B->xyz|xyw|xyt"

};

int ruleCount = 2;

printf("Grammar after removing left factoring:\n");

eliminateLeftFactoring(rules, ruleCount);

return 0;

}

OUTPUT:

