**Aim**

To develop a **C program** to eliminate **left factoring** from a given **context-free grammar (CFG)**.

**C Program to Eliminate Left Factoring**

c

CopyEdit

#include <stdio.h>

#include <string.h>

#define MAX 10

char productions[MAX][MAX]; // Stores input grammar

char newProductions[MAX][MAX]; // Stores transformed grammar

int numProductions; // Number of productions

// Function to eliminate left factoring

void eliminateLeftFactoring() {

int newProdCount = 0;

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

char nonTerminal = productions[i][0]; // Get LHS (e.g., 'S')

char commonPrefix[MAX] = "";

char suffix[MAX][MAX];

int suffixCount = 0;

// Extract RHS

char \*rhs = strchr(productions[i], '>') + 1;

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

while (token != NULL) {

if (suffixCount == 0) {

strcpy(commonPrefix, token);

} else {

int j;

for (j = 0; commonPrefix[j] && token[j] && commonPrefix[j] == token[j]; j++);

commonPrefix[j] = '\0';

}

strcpy(suffix[suffixCount++], token);

token = strtok(NULL, "|");

}

if (strlen(commonPrefix) > 0 && suffixCount > 1) {

char newNonTerminal = nonTerminal + '\'';

sprintf(newProductions[newProdCount++], "%c->%s%c", nonTerminal, commonPrefix, newNonTerminal);

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

if (strcmp(suffix[j] + strlen(commonPrefix), "") == 0) {

sprintf(newProductions[newProdCount++], "%c->ε", newNonTerminal);

} else {

sprintf(newProductions[newProdCount++], "%c->%s", newNonTerminal, suffix[j] + strlen(commonPrefix));

}

}

} else {

strcpy(newProductions[newProdCount++], productions[i]);

}

}

numProductions = newProdCount;

}

// Function to print the updated grammar

void printNewGrammar() {

printf("\nGrammar after eliminating left factoring:\n");

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

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

}

}

int main() {

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

scanf("%d", &numProductions);

getchar(); // Consume newline

printf("Enter the productions (e.g., S->ab|ac|d):\n");

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

fgets(productions[i], MAX, stdin);

productions[i][strcspn(productions[i], "\n")] = '\0'; // Remove newline

}

eliminateLeftFactoring();

printNewGrammar();

return 0;

}

**Example Input and Output**

**Input**

rust

CopyEdit

Enter the number of productions: 1

Enter the productions (e.g., S->ab|ac|d):

S->ab|ac|d

**Output**

vbnet

CopyEdit

Grammar after eliminating left factoring:

S->aS'|d

S'->b|c