**EX 7:** **Implement a C program to eliminate left factoring.**

**Aim:**

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

**Algorithm**

1. Read the grammar rules.
2. Identify productions with common prefixes.
3. Factor out the common prefixes and introduce new non-terminals.
4. Generate and display the modified grammar without left factoring.

**Code:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX 10

void leftFactor(char nonTerminal, char \*productions[], int count) {

char prefix[MAX], newSymbol, newProduction[MAX][MAX];

int i, j, k, prefixLength = 0, flag = 1;

for (i = 0; i < strlen(productions[0]); i++) {

char ch = productions[0][i];

for (j = 1; j < count; j++) {

if (productions[j][i] != ch) {

flag = 0;

break;

}

}

if (!flag) break;

prefix[prefixLength++] = ch;

}

prefix[prefixLength] = '\0';

if (prefixLength == 0) {

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

for (i = 0; i < count; i++) {

printf("%s", productions[i]);

if (i < count - 1) printf(" | ");

}

printf("\n");

return;

}

newSymbol = 'X';

printf("%c -> %s%c\n", nonTerminal, prefix, newSymbol);

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

for (i = 0; i < count; i++) {

if (strlen(productions[i]) == prefixLength) {

printf("ε"); // If only prefix, use epsilon

} else {

printf("%s", productions[i] + prefixLength);

}

if (i < count - 1) printf(" | ");

}

printf("\n");

}

int main() {

char nonTerminal;

int i, count;

char \*productions[MAX];

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

scanf(" %c", &nonTerminal);

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

scanf("%d", &count);

for (i = 0; i < count; i++) {

productions[i] = (char \*)malloc(MAX \* sizeof(char));

printf("Enter production %d: ", i + 1);

scanf("%s", productions[i]);

}

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

leftFactor(nonTerminal, productions, count);

for (i = 0; i < count; i++) {

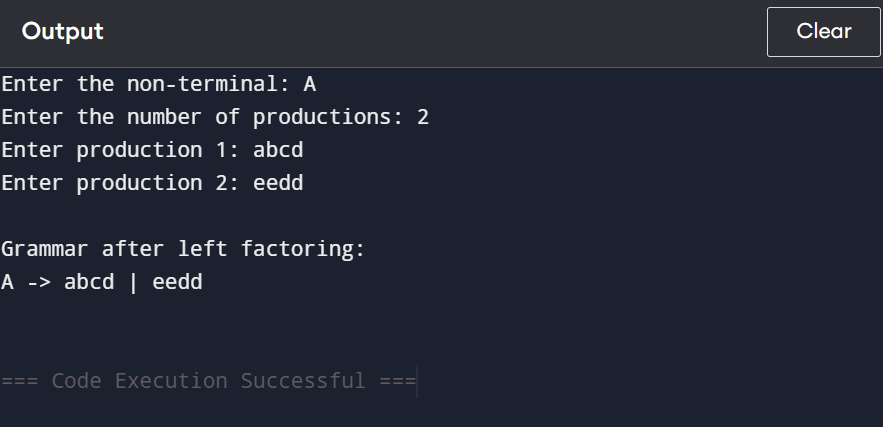
free(productions[i]);

}

return 0;

}

**Output:**

****