Write a C program for implementing a Lexical Analyzer to Scan and Count the number of characters, words, and lines in a file.

**Code:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX 10

typedef struct {

char nonTerminal;

char production[MAX][MAX];

int prodCount;

} Grammar;

Grammar grammar[MAX];

int grammarCount = 0;

void computeTrailing(char symbol, char trailingSet[MAX], int \*index) {

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

if (grammar[i].nonTerminal == symbol) {

for (int j = 0; j < grammar[i].prodCount; j++) {

char \*prod = grammar[i].production[j];

int len = strlen(prod);

if (len > 0) {

char lastChar = prod[len - 1];

if (!isupper(lastChar)) { // If last char is a terminal

trailingSet[(\*index)++] = lastChar;

} else { // If last char is a non-terminal

computeTrailing(lastChar, trailingSet, index);

}

}

}

}

}

}

void printTrailing() {

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

char trailingSet[MAX] = {0};

int index = 0;

computeTrailing(grammar[i].nonTerminal, trailingSet, &index);

printf("TRAILING(%c) = { ", grammar[i].nonTerminal);

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

printf("%c ", trailingSet[j]);

}

printf("}\n");

}

}

int main() {

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

scanf("%d", &grammarCount);

getchar(); // Consume newline

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

printf("Enter production (e.g., E->E+T): ");

char input[MAX];

fgets(input, MAX, stdin);

input[strcspn(input, "\n")] = 0; // Remove newline

grammar[i].nonTerminal = input[0];

strcpy(grammar[i].production[0], input + 3);

grammar[i].prodCount = 1;

}

printTrailing();

return 0;

}

**OUTPUT:**

