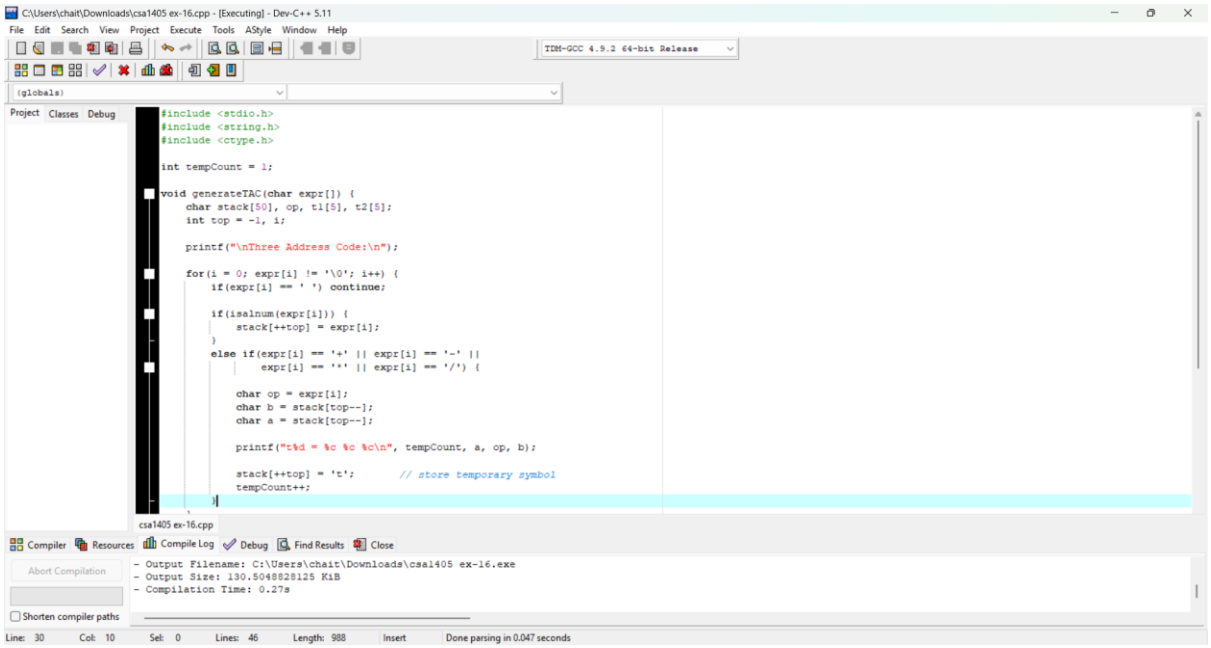
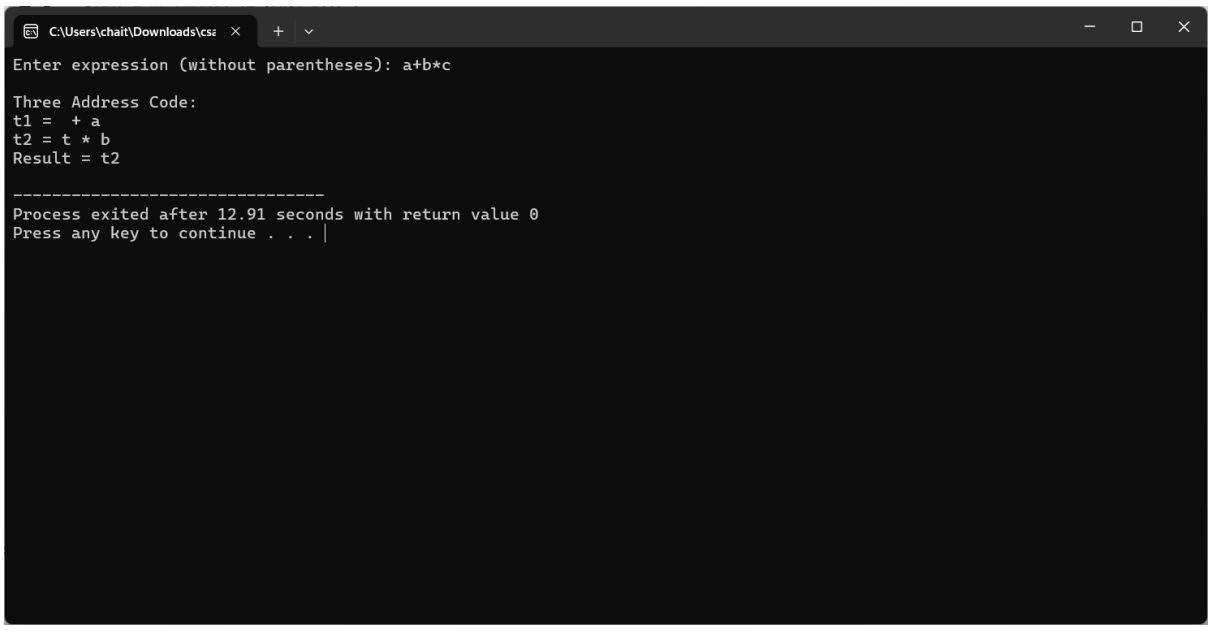
Exp. No. 16

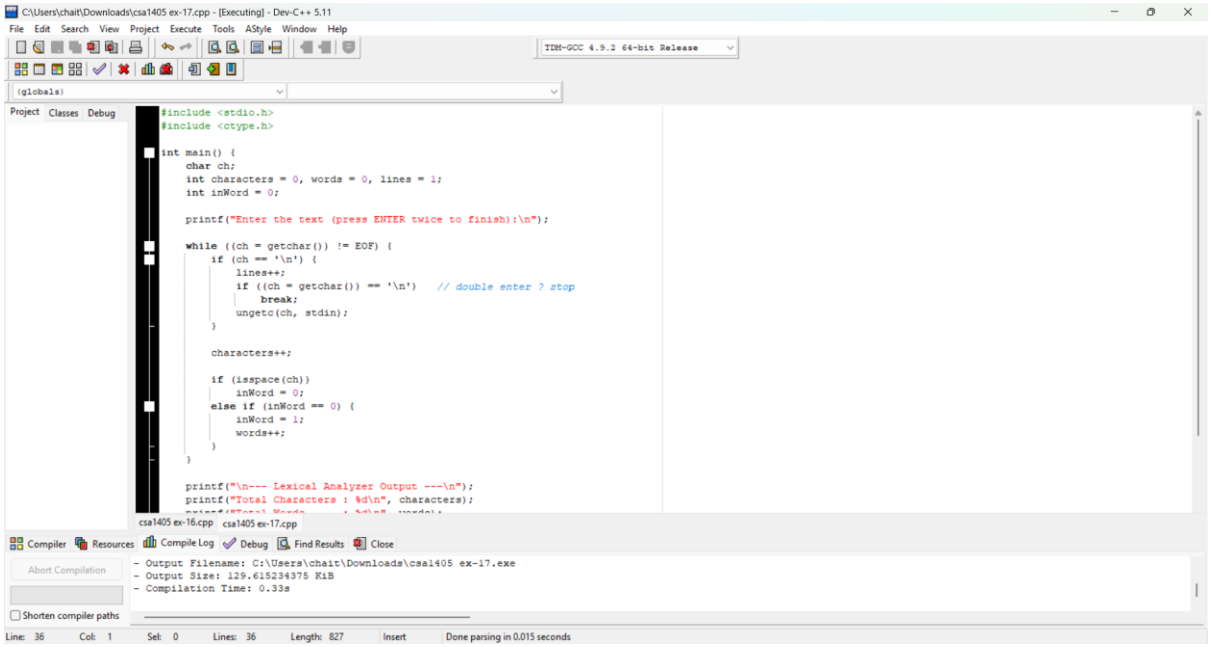
Write a C Program to Generate the Three address code representation for the given input statement.

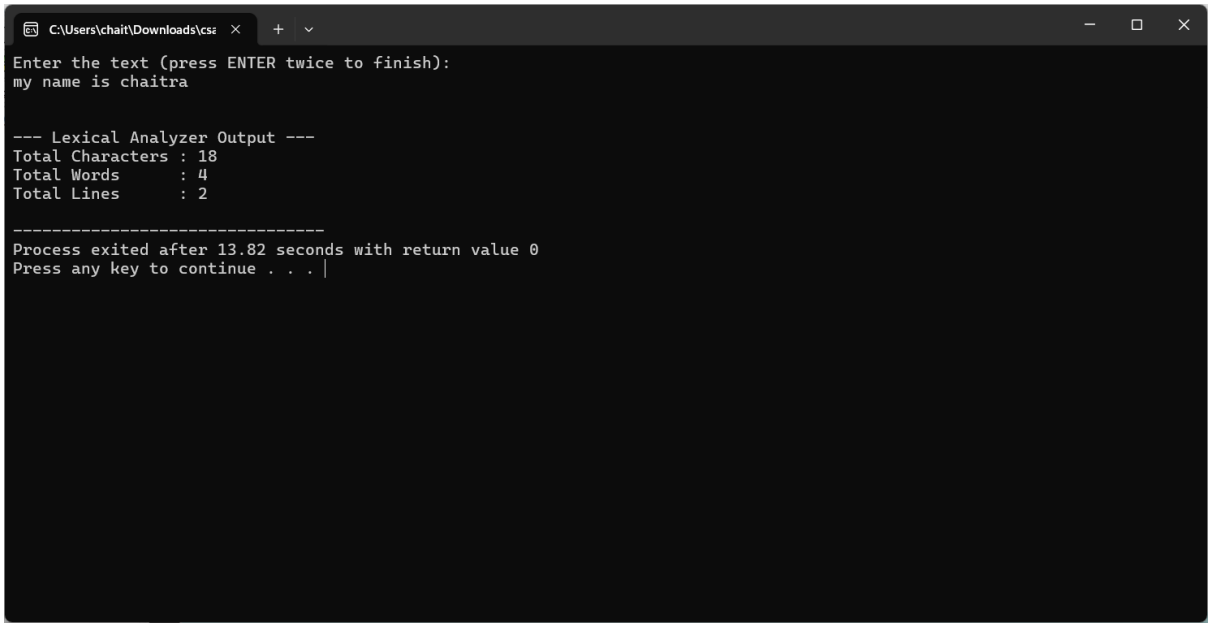




Exp. No. 17

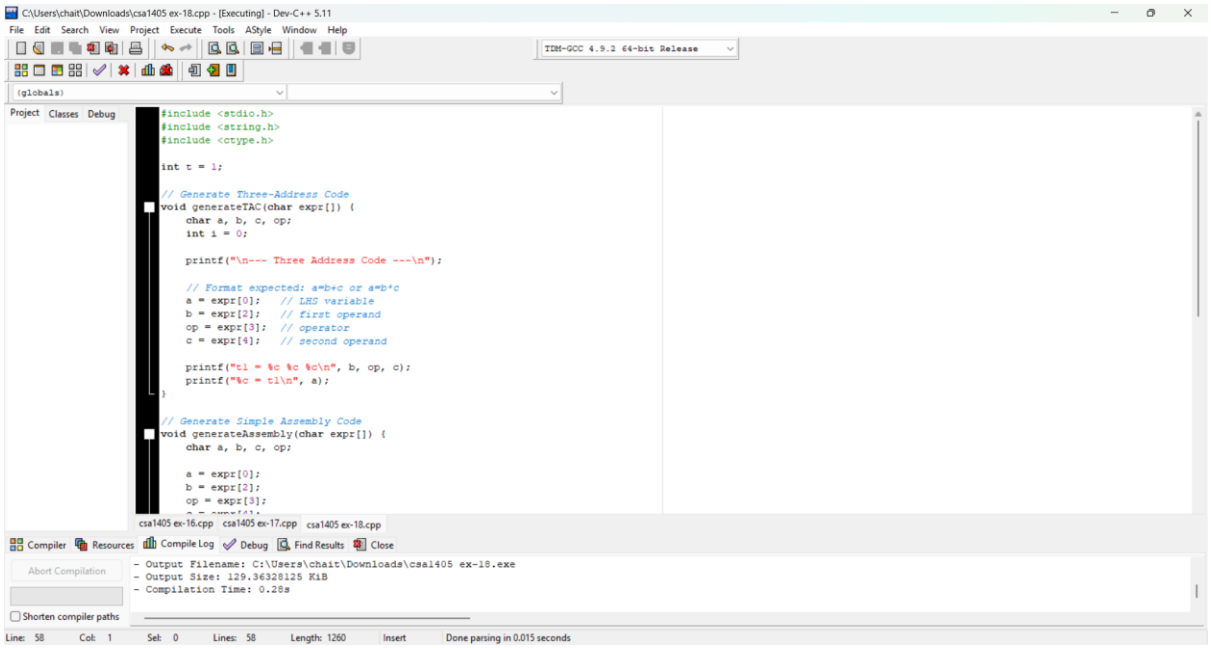
Write a C program for implementing a Lexical Analyzer to Scan and Count the number of characters, words, and lines in a file.

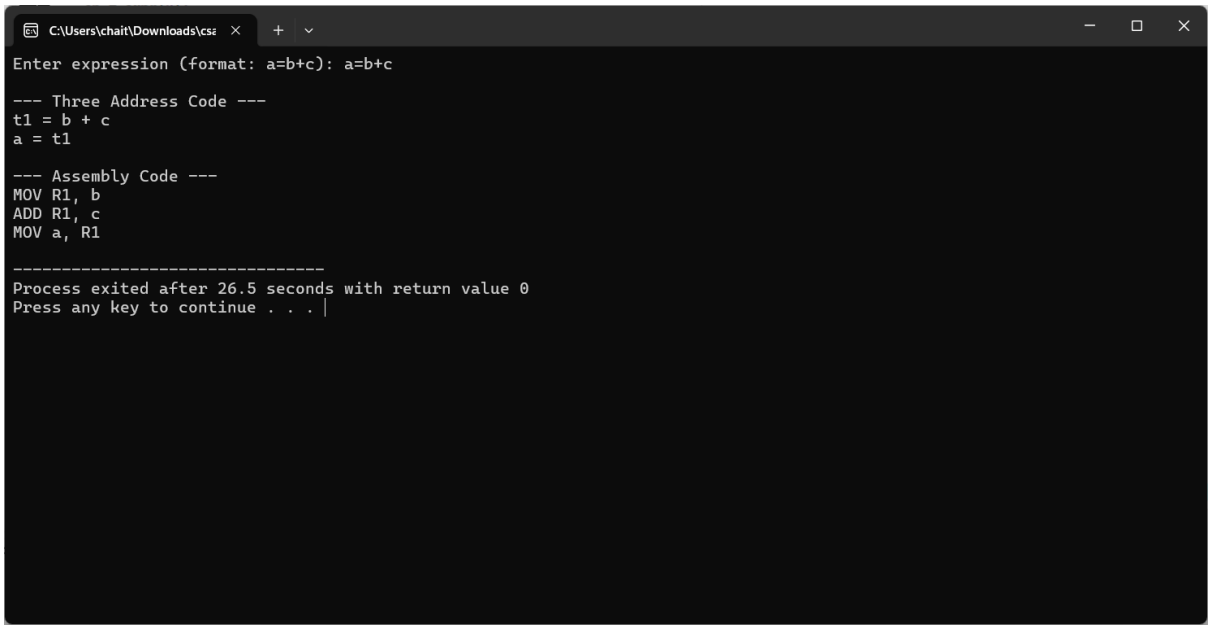




Exp. No. 18

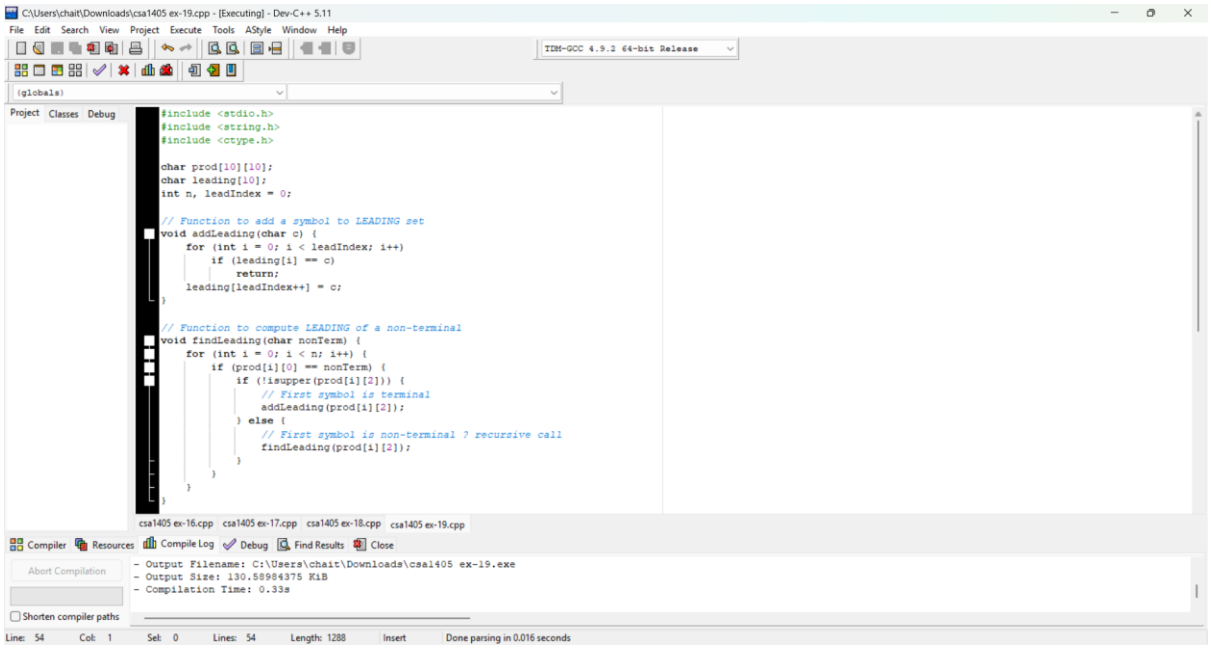
Write a C program to implement the back end of the compiler.





Exp. No. 19

Write a C program to compute LEADING( ) – operator precedence parser for the given grammar



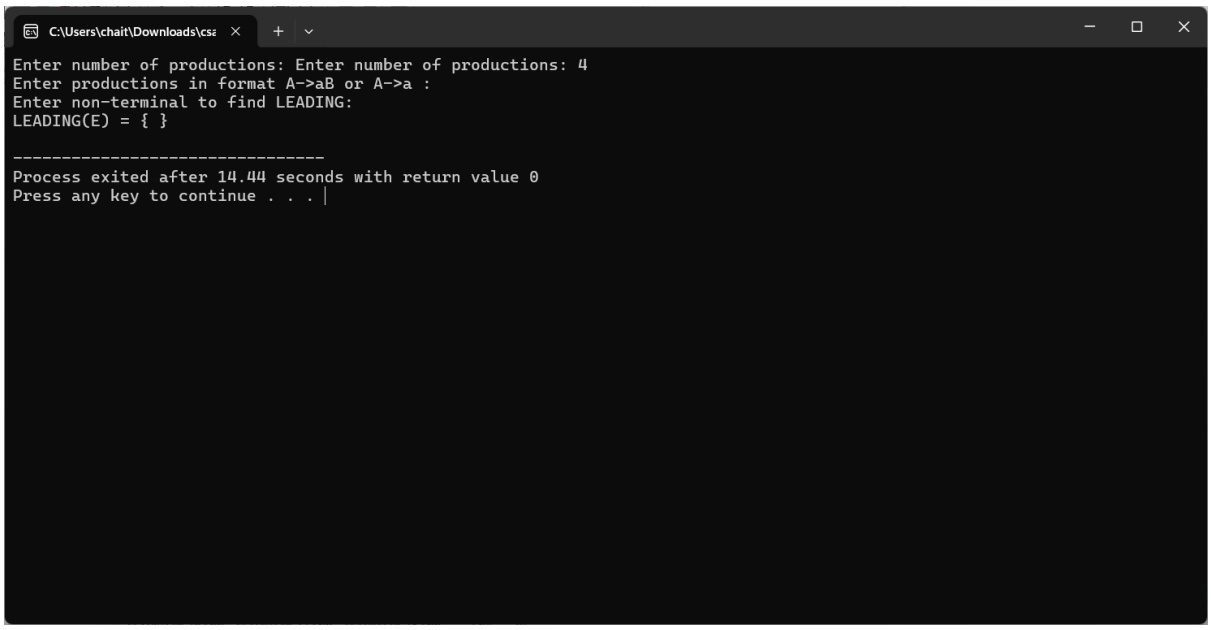
```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

char prod[10][10];
char leading[10];
int n, leadIndex = 0;

// Function to add a symbol to LEADING set
void addLeading(char c) {
    for (int i = 0; i < leadIndex; i++)
        if (leading[i] == c)
            return;
    leading[leadIndex++] = c;
}

// Function to compute LEADING of a non-terminal
void findLeading(char nonTerm) {
    for (int i = 0; i < n; i++) {
        if (prod[i][0] == nonTerm) {
            if (!isupper(prod[i][2])) {
                // First symbol is terminal
                addLeading(prod[i][2]);
            } else {
                // First symbol is non-terminal ? recursive call
                findLeading(prod[i][2]);
            }
        }
    }
}
```

```
- Output Filename: C:\Users\chait\Downloads\csa1405 ex-19.exe
- Output Size: 130.58984375 KiB
- Compilation Time: 0.33s
```

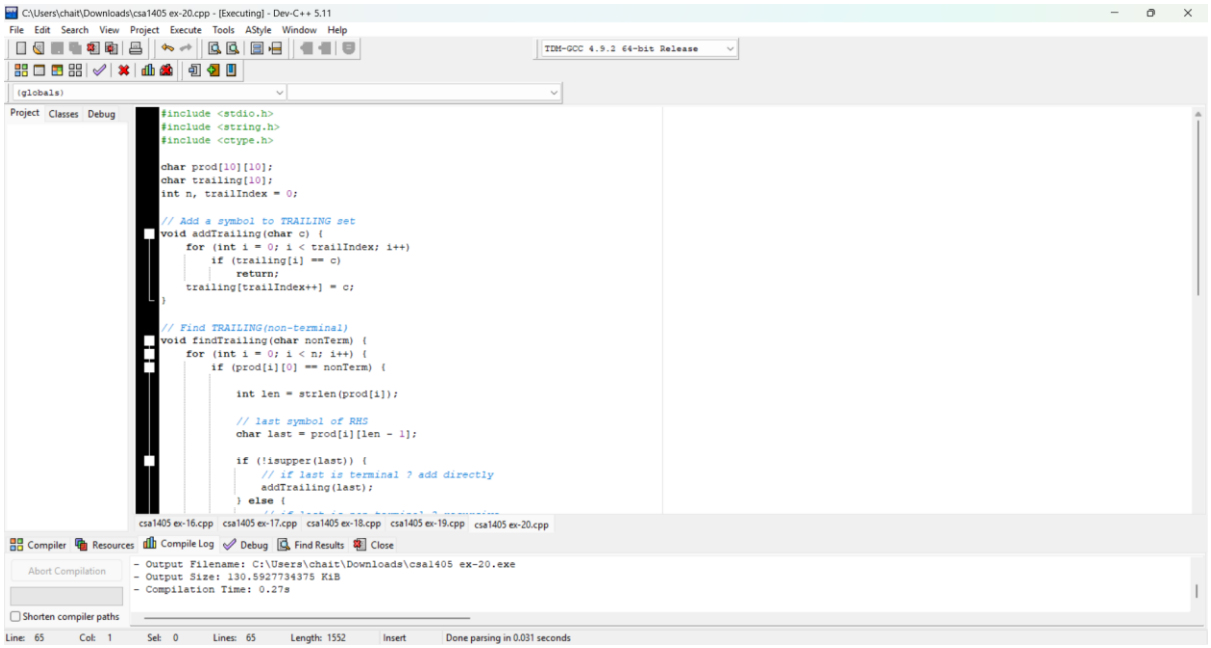


```
Enter number of productions: Enter number of productions: 4
Enter productions in format A->aB or A->a :
Enter non-terminal to find LEADING:
LEADING(E) = { }

--------------------------------
Process exited after 14.44 seconds with return value 0
Press any key to continue . . .
```

Exp. No. 20

Write a C program to compute TRAILING( ) – operator precedence parser for the given grammar



```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

char prod[10][10];
char trailing[10];
int n, trailIndex = 0;

// Add a symbol to TRAILING set
void addTrailing(char c) {
    for (int i = 0; i < trailIndex; i++)
        if (trailing[i] == c)
            return;
    trailing[trailIndex++] = c;
}

// Find TRAILING(non-terminal)
void findTrailing(char nonTerm) {
    for (int i = 0; i < n; i++) {
        if (prod[i][0] == nonTerm) {

            int len = strlen(prod[i]);

            // last symbol of RHS
            char last = prod[i][len - 1];

            if (!isupper(last)) {
                // if last is terminal ? add directly
                addTrailing(last);
            } else {
```

```
Enter number of productions: Enter number of productions: 4
Enter productions (A->aB or A->ab or A->B):
Enter non-terminal to find TRAILING:
TRAILING(E) = { }

--------------------------------
Process exited after 21.11 seconds with return value 0
Press any key to continue . . .
```