

main.c



Run

Output

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdbool.h>
4
5 #define MAX_PROD 10
6 #define MAX_SYMBOLS 10
7
8 char productions[MAX_PROD][MAX_SYMBOLS];
9 int num_productions;
10
11 // Function to check if a symbol is a terminal
12 bool isTerminal(char symbol) {
13     return !(symbol >= 'A' && symbol <= 'Z');
14 }
15
16 // Function to compute the LEADING set for a non-terminal
17 void computeLeading(char non_terminal, char leading_set[], int *leading_count) {
18     for (int i = 0; i < num_productions; i++) {
19         if (productions[i][0] == non_terminal) {
20             char first_symbol = productions[i][3]; // First symbol after "->"
21             if (isTerminal(first_symbol)) {
22                 // If the first symbol is a terminal, add it to the LEADING set
23                 leading_set[(*leading_count)++] = first_symbol;
24             } else {
25                 // If the first symbol is a non-terminal, recursively compute its LEADING set
26                 computeLeading(first_symbol, leading_set, leading_count);
27             }
28         }
29     }
30 }
31
32 int main() {
33     // Example grammar
34     strcpy(productions[0], "E->E+T");
35     strcpy(productions[1], "E->T");
36     strcpy(productions[2], "T->T*F");
37     strcpy(productions[3], "T->F");
```

Segmentation fault

=== Code Exited With Errors ===

```

strcpy(productions[4], "F->E");
strcpy(productions[5], "F->id");
num_productions = 6;

// Compute LEADING set for non-terminal E
char leading_set_E[MAX_SYMBOLS];
int leading_count_E = 0;
computeLeading('E', leading_set_E, &leading_count_E);

// Print the LEADING set for E
printf("LEADING(E) = { ");
for (int i = 0; i < leading_count_E; i++) {
    printf("%c ", leading_set_E[i]);
}
printf("}\n");

// Compute LEADING set for non-terminal T
char leading_set_T[MAX_SYMBOLS];
int leading_count_T = 0;
computeLeading('T', leading_set_T, &leading_count_T);

// Print the LEADING set for T
printf("LEADING(T) = { ");
for (int i = 0; i < leading_count_T; i++) {
    printf("%c ", leading_set_T[i]);
}
printf("}\n");

// Compute LEADING set for non-terminal F
char leading_set_F[MAX_SYMBOLS];
int leading_count_F = 0;
computeLeading('F', leading_set_F, &leading_count_F);

// Print the LEADING set for F
printf("LEADING(F) = { ");

```

Segmentation fault

=== Code Exited With Errors ===

main.c



Share

Run

Output

```
44 int leading_count_E = 0;
45 computeLeading('E', leading_set_E, &leading_count_E);
46
47 // Print the LEADING set for E
48 printf("LEADING(E) = { ");
49 for (int i = 0; i < leading_count_E; i++) {
50     printf("%c ", leading_set_E[i]);
51 }
52 printf("\n");
53
54 // Compute LEADING set for non-terminal T
55 char leading_set_T[MAX_SYMBOLS];
56 int leading_count_T = 0;
57 computeLeading('T', leading_set_T, &leading_count_T);
58
59 // Print the LEADING set for T
60 printf("LEADING(T) = { ");
61 for (int i = 0; i < leading_count_T; i++) {
62     printf("%c ", leading_set_T[i]);
63 }
64 printf("\n");
65
66 // Compute LEADING set for non-terminal F
67 char leading_set_F[MAX_SYMBOLS];
68 int leading_count_F = 0;
69 computeLeading('F', leading_set_F, &leading_count_F);
70
71 // Print the LEADING set for F
72 printf("LEADING(F) = { ");
73 for (int i = 0; i < leading_count_F; i++) {
74     printf("%c ", leading_set_F[i]);
75 }
76 printf("\n");
77
78 return 0;
79 }
```

Segmentation fault

=== Code Exited With Errors ===