Experiment -19

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

E → E + T | T

T → T * F | F

F → (E) | id

Program:

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#define MAX 10
char nonTerminals[] = {'E', 'T', 'F'};
int nNT = 3;
char productions[10][20] = {
    "E=E+T",
    "E=T",
    "T=T*F",
    "T=F",
    "F=(E)",
    "F=i"
};
int nProd = 6;
char leading[10][10];
int leadCount[10] = {0};
void addLeading(int ntIndex, char symbol) {
    for (int i = 0; i < leadCount[ntIndex]; i++) {
        if (leading[ntIndex][i] == symbol)
            return;
    }
    leading[ntIndex][leadCount[ntIndex]++] = symbol;
```

```
}
int getNTindex(char A) {
    for (int i = 0; i < nNT; i++)
        if (nonTerminals[i] == A)
            return i;
    return -1;
}
int main() {
    int updated = 1;
    printf("\nGiven Grammar:\n");
    for (int i = 0; i < nProd; i++)
        printf("  %s\n", productions[i]);
    while (updated) {
        updated = 0;
        for (int p = 0; p < nProd; p++) {
            char A = productions[p][0];      // LHS Non-terminal
            int indexA = getNTindex(A);
            char *rhs = strchr(productions[p], '=') + 1;
            if (ispunct(rhs[0]) || islower(rhs[0]) || rhs[0] == 'i') {
                int before = leadCount[indexA];
                addLeading(indexA, rhs[0]);
                if (leadCount[indexA] != before) updated = 1;
            }
            if (isupper(rhs[0])) {
                int indexB = getNTindex(rhs[0]);
                for (int k = 0; k < leadCount[indexB]; k++) {
                    int before = leadCount[indexA];
                    addLeading(indexA, leading[indexB][k]);
                    if (leadCount[indexA] != before) updated = 1;} }}}
    printf("\n\nLEADING Sets:\n");
```

```
    for (int i = 0; i < nNT; i++) {
        printf("\nLEADING(%c) = { ", nonTerminals[i]);
        for (int j = 0; j < leadCount[i]; j++)
            printf("%c ", leading[i][j]);
        printf("}");
    }
    printf("\n");
    return 0;
}
```

Output: