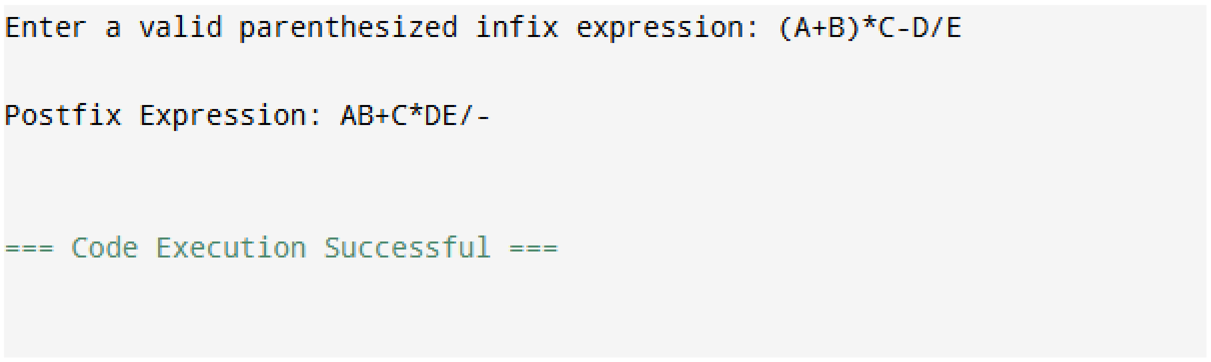
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define SIZE 100

char stack[SIZE];
int top = -1;

void push(char c) {
    if (top == SIZE - 1)
        printf("Stack Overflow\n");
    else
        stack[++top] = c;
}

char pop() {
    if (top == -1)
        return -1;
    else
        return stack[top--];
}

int precedence(char c) {
    if (c == '(')
        return 0;
    if (c == '+' || c == '-')
        return 1;
    if (c == '*' || c == '/')
        return 2;
    return 0;
}

void infixToPostfix(char infix[]) {
    char postfix[SIZE];
    int i, j = 0;
    char x;

    for (i = 0; i < strlen(infix); i++) {
        char symbol = infix[i];

        if (isalnum(symbol)) {
            postfix[j++] = symbol; // operand → directly to output
        }
        else if (symbol == '(') {
            push(symbol);
        }
        else if (symbol == ')') {
            while ((x = pop()) != '(')
                postfix[j++] = x;
        }
        else { // operator
            while (precedence(stack[top]) >= precedence(symbol))
                postfix[j++] = pop();
            push(symbol);
        }
    }

    while (top != -1)
        postfix[j++] = pop();

    postfix[j] = '\0';
    printf("\nPostfix Expression: %s\n", postfix);
}

int main() {
    char infix[SIZE];

    printf("Enter a valid parenthesized infix expression: ");
    scanf("%s", infix);

    infixToPostfix(infix);

    return 0;
}
```



```
Enter a valid parenthesized infix expression: (A+B)*C-D/E

Postfix Expression: AB+C*DE/-


=== Code Execution Successful ===
```