LAB 5 - (1)Qns - compare_expressions

A software engineer tasked with developing a program that checks if two mathematical expressions are similar. These expressions consist of lowercase alphabets, addition (+), subtraction (-), and parentheses (). The goal is to determine if the two expressions, when simplified, are equivalent. The expressions may contain brackets and involve various arithmetic operations. Basically compare expressions. For example, given the expressions -(a+b)-c and -a-b-c are same.

Note: Utilize stack Data Structure. Utilize the provided boilerplate code or develop your own, ensuring it aligns with the given input and output formats. Minimize extraneous print statements; your output should match the specified format.

Input Format

First Line has string 1 with brackets. Second Line has string 2 without brackets

Constraints

length of string less than 50

Output Format

Prints YES or NO

Sample Input 0

- `(a+b)-c`
  `-a-b-c`

Sample Output 0

`YES`

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <stdlib.h>

#define MAX_CHAR 26

bool adjSign(char *s, int i) {
```

```
    if (i == 0)
        return true;
    if (s[i - 1] == '-')
        return false;
    return true;
}

void eval(char *s, int v[], bool add) {
    bool stk[1000];
    int top = -1;
    stk[++top] = true;
    int i = 0;
    while (s[i] != '\0') {
        if (s[i] == '+' || s[i] == '-') {
            i++;
            continue;
        }
        if (s[i] == '(') {
            bool globalSign;
            if (adjSign(s, i))
                globalSign = stk[top];
            else
                globalSign = !stk[top];

            top++;
            stk[top] = globalSign;
        }
        else if (s[i] == ')')
            top--;

        else {
            bool localSign = adjSign(s, i);
            int sign = (stk[top]) ? (localSign ? (add ? 1 : -1) : (add ? -1 : 1)) :
                                    (localSign ? (add ? -1 : 1) : (add ? 1 : -1));
            v[s[i] - 'a'] += sign;
        }
        i++;
    }
}

bool areSame(char *expr1, char *expr2) {
    int v[MAX_CHAR] = {0};
    eval(expr1, v, true);
    eval(expr2, v, false);
    for (int i = 0; i < MAX_CHAR; i++)
        if (v[i] != 0)
            return false;
    return true;
}

int main() {
    char expr1[100]; // = "-(a+b+c)";
    fgets(expr1, sizeof(expr1), stdin);
    expr1[strcspn(expr1, "\n")] = '\0';

    char expr2[100]; // = "-a-b-c";
```

```
    fgets(expr2, sizeof(expr2), stdin);
    expr2[strcspn(expr2, "\n")] = '\0';
    if (areSame(expr1, expr2))
        printf("YES\n");
    else
        printf("NO\n");
    return 0;
}
```