Balanced Brackets

```c
#include <assert.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char* ltrim(char*);
char* rtrim(char*);

int parse_int(char*);

/*
 * Complete the 'isBalanced' function below.
 *
 * The function is expected to return a STRING.
 * The function accepts STRING s as parameter.
 */

/*
 * To return the string from the function, you should either do static allocation or dynamic allocation
 *
 * For example,
 * char* return_string_using_static_allocation() {
 *     static char s[] = "static allocation of string";
 *
 *     return s;
 * }
 *
 * char* return_string_using_dynamic_allocation() {
 *     char* s = malloc(100 * sizeof(char));
 *
 *     s = "dynamic allocation of string";
 *
 *     return s;
 * }
 *
```

```
 */
char* isBalanced(char* s) {
    int len=strlen(s);
    char* stack=malloc(len);
    int top=-1;
    for(int i=0;i<len;i++){
        char c=s[i];
        if(c=='('||c=='['||c=='{'){
            stack[++top]=c;
        } else {
            if(top==-1){
                free(stack);
                return "NO";
            }
            char open=stack[top--];
            if((c==')' && open!='(') ||
               (c==']' && open!='[') ||
               (c=='}' && open!='{')){
                free(stack);
                return "NO";
            }
        }
    }
    if(top==-1){
        free(stack);
        return "YES";
    } else {
        free(stack);
        return "NO";
    }
}

int main()
{
    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

    int t = parse int(ltrim(rtrim(readline())));

    for (int t itr = 0; t itr < t; t itr++) {
        char* s = readline();

        char* result = isBalanced(s);

        fprintf(fptr, "%s\n", result);
    }
```

```
    fclose(fptr);

    return 0;
}

char* readline() {
    size_t alloc_length = 1024;
    size_t data_length = 0;

    char* data = malloc(alloc_length);

    while (true) {
        char* cursor = data + data_length;
        char* line = fgets(cursor, alloc_length - data_length,
stdin);

        if (!line) {
            break;
        }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_length -
1] == '\n') {
            break;
        }

        alloc_length <<= 1;

        data = realloc(data, alloc_length);

        if (!data) {
            data = '\0';

            break;
        }
    }

    if (data[data length - 1] == '\n') {
        data[data length - 1] = '\0';

        data = realloc(data, data_length);

        if (!data) {
            data = '\0';
        }
```

```
    } else {
        data = realloc(data, data_length + 1);

        if (!data) {
            data = '\0';
        } else {
            data[data_length] = '\0';
        }
    }

    return data;
}

char* ltrim(char* str) {
    if (!str) {
        return '\0';
    }

    if (!*str) {
        return str;
    }

    while (*str != '\0' && isspace(*str)) {
        str++;
    }

    return str;
}

char* rtrim(char* str) {
    if (!str) {
        return '\0';
    }

    if (!*str) {
        return str;
    }

    char* end = str + strlen(str) - 1;

    while (end >= str && isspace(*end)) {
        end--;
    }

    *(end + 1) = '\0';
```

```
    return str;
}

int parse_int(char* str) {
    char* endptr;
    int value = strtol(str, &endptr, 10);

    if (endptr == str || *endptr != '\0') {
        exit(EXIT_FAILURE);
    }

    return value;
}
```