Setiawan Muhammad
1203230016
IF 03-01

```c
#include <stdio.h>  // Library standar untuk fungsi input-output
#include <string.h> // Library untuk fungsi-fungsi terkait string
#include <stdlib.h> // Library untuk fungsi alokasi dan dealokasi memori dinamis

// Deklarasi struktur Stack untuk merepresentasikan stack tanda kurung
typedef struct Stack {
    char* kurung; // Pointer ke array karakter untuk menyimpan tanda kurung
    int top;      // Indeks atas dari stack
    int size;     // Ukuran maksimal stack
} Stack;

// Fungsi untuk membuat stack baru dengan ukuran tertentu
Stack* buatStack(int size) {
    Stack* stack = (Stack*)malloc(sizeof(Stack)); // Alokasi memori untuk stack
    stack->kurung = (char*)malloc(size * sizeof(char)); // Alokasi memori untuk array tanda kurung
    stack->top = -1; // Inisialisasi indeks atas stack
    stack->size = size; // Menyimpan ukuran maksimal stack

    return stack; // Mengembalikan pointer ke stack yang telah dibuat
}

// Fungsi untuk mengecek apakah stack kosong atau tidak
int isEmpty(Stack* stack) {
    return stack->top == -1; // Mengembalikan 1 jika stack kosong, 0 jika tidak kosong
```

```
}

// Fungsi untuk menambahkan elemen ke dalam stack
void push(Stack* stack, char kurawa) {
    stack->kurung[++stack->top] = kurawa; // Menambahkan karakter ke dalam
stack dan menggeser indeks atas
}

// Fungsi untuk menghapus dan mengembalikan elemen teratas dari stack
char pop(Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack kosong\n");
        return '\0'; // Jika stack kosong, kembalikan karakter null
    }
    return stack->kurung[stack->top--]; // Menghapus dan mengembalikan
elemen teratas stack
}

// Fungsi untuk mengecek keseimbangan tanda kurung dalam string
char* isBalanced(char* kondisi) {
    Stack* stack = buatStack(strlen(kondisi)); // Membuat stack baru
dengan ukuran panjang string input
    for (int i = 0; kondisi[i] != '\0'; i++) { // Iterasi untuk setiap
karakter dalam string input
        if (kondisi[i] == '(' || kondisi[i] == '{' || kondisi[i] == '[')
{ // Jika tanda kurung pembuka ditemukan
            push(stack, kondisi[i]); // Masukkan ke dalam stack
        } else if (kondisi[i] == ')' || kondisi[i] == '}' || kondisi[i] ==
']') { // Jika tanda kurung penutup ditemukan
            if (isEmpty(stack)) { // Jika stack kosong, tidak seimbang
                free(stack->kurung);
                free(stack);
                return "NO";
            } else {
```

```
                char popped = pop(stack); // Ambil tanda kurung pembuka
teratas dari stack
                if ((kondisi[i] == ')' && popped != '(') || // Jika tidak
seimbang, kembalikan "NO"
                    (kondisi[i] == '}' && popped != '{') ||
                    (kondisi[i] == ']' && popped != '[')) {
                    free(stack->kurung);
                    free(stack);
                    return "NO";
                }
            }
        }
    }

    if (isEmpty(stack)) { // Jika stack tidak kosong pada akhir iterasi,
seimbang
        free(stack->kurung);
        free(stack);
        return "YES";
    }

    free(stack->kurung);
    free(stack);
    return "NO"; // Jika tidak kosong pada akhir iterasi, tidak seimbang
}

// Fungsi utama
int main() {
    char kondisi[100];
    printf("Silahkan masukkan tanda kurung : ");
    scanf("%[^\n]s", kondisi); // Membaca string input dari pengguna

    char* hasil = isBalanced(kondisi); // Memanggil fungsi isBalanced
untuk mengecek keseimbangan tanda kurung
```

```
    printf("%s", hasil);


    return 0;
}
```