NAMA : ELDHIEN ANGGI DERMAWAN RAMBE
NIM : 1203230109

TUGAS OTS

1. SOURCODE

```c
#include <stdio.h>

#include <stdlib.h>

//berfungsi untuk membuat sebuah tipe data baru
typedef struct Stack {
    char data;//variabel menyimpan data
    struct Stack* next;
} Stack;

void push(Stack** top, char data) {
    Stack* new_node = (Stack*) malloc(sizeof(Stack));//
    if (!new_node) return;
    new_node->data = data;
    new_node->next = (*top);
    (*top) = new_node;
}
//fungsi untuk memeriksa  apakah stuck kosong
int isEmpty(Stack* top) {
    return top == NULL;
}

char pop(Stack** top) {
    char popped;
    Stack* temp;
    if (isEmpty(*top)) return '\0';
    temp = *top;
    *top = (*top)->next;
    popped = temp->data;
    free(temp);
    return popped;
}

char peek(Stack* top) {
    if (isEmpty(top)) return '\0';
    return top->data;
}
// memeriksa apakah urutan tanda kurung sesuai
int isMatchingPair(char character1, char character2) {
   if (character1 == '(' && character2 == ')') return 1;
   else if (character1 == '{' && character2 == '}') return 1;
   else if (character1 == '[' && character2 == ']') return 1;
   else return 0;
```

```
}

int isBalanced(char exp[]) {
    int i = 0;
    Stack* stack = NULL;
    while (exp[i]) {
        if (exp[i] == '{' || exp[i] == '(' || exp[i] == '[')
            push(&stack, exp[i]);
        if (exp[i] == '}' || exp[i] == ')' || exp[i] == ']') {
            if (stack == NULL) return 0;
            else if (!isMatchingPair(pop(&stack), exp[i])) return 0;
        }
        i++;
    }
    if (stack == NULL) return 1;
    else return 0;
}

int main() {
    char exp[100];
    printf("Masukkan urutan tanda kurung: ");
    scanf("%s", exp);
    if (isBalanced(exp))
        printf("YES\n");
    else
        printf("NO\n");
    return 0;
}
```

2. PENJELASAN

```
#include <stdio.h>

#include <stdlib.h>
```

`<stdio.h>` berfungsi sebagai mendefenisikan deader file untuk menerima funsi input dan output dalam Bahasa C

`<stdlib.h>` berfungsi mendefenisikan header file untuk fungsi umum, seperti alokasi memori dan fungsi konverssi

```
typedef struct Stack {
    char data;
    struct Stack* next;
} Stack;
```

`typedef struct` berfungsi membuat sebuah tipe data baru `Stack` yang merupakan tipe data baru untuk menyimpan elemen stack

`char data` variable untuk menyimpan data dalam setiap stack

`struct Stack* next` pointer yang menunjuk kedalam lanjutan stack

```
void push(Stack** top, char data) {
    Stack* new_node = (Stack*) malloc(sizeof(Stack));
    if (!new_node) return;
    new_node->data = data;
    new_node->next = (*top);
    (*top) = new_node;
```

`void push(Stack** top, char data` berfungsi untuk menambahkan elemen kedalam stack

`Stack* new_node = (Stack*)` berfungsi untuk alokasi nide yang baru kedalam stack `malloc`

`new_node->data = data;` ini adalah Langkah pencegahan untuk memastikan bahwa alokasi memori berhaasil.

`new_node->next = (*top);` iini berfungsi untuk menunjuk nebo baru ke elemen teratas stack saat ini

```
int isEmpty(Stack* top) {
    return top == NULL;
```

`int isEmpty(Stack* top` berfungsi untuk memeriksa apakah apakah stack kosong

```
char pop(Stack** top) {
    char popped;
    Stack* temp;
    if (isEmpty(*top)) return '\0';
    temp = *top;
    *top = (*top)->next;
    popped = temp->data;
    free(temp);
    return popped;
```

`char pop(Stack** top)` baris ini berfungsi untuk menghapus stack yang teratas dan mengembalikaan stack teraatas dari elemen sebelumnya

```
char peek(Stack* top) {
    if (isEmpty(top)) return '\0';
    return top->data;
```

`char peek(Stack* top` baris ini berfungsi untuk melihat nilai dari elemen teratas tampa menghapusnya

```
int isMatchingPair(char character1, char character2) {
   if (character1 == '(' && character2 == ')') return 1;
   else if (character1 == '{' && character2 == '}') return 1;
   else if (character1 == '[' && character2 == ']') return 1;
   else return 0;
```

pada baris ini akan memeriksa apakah pasangan tanda kurung sudah sesuai, dan pada baris ini memiliki 2 charakter sabagai parameter `character1` dan `character2` yang mewakili pasangan karakter yang akan di periksa pakah sesuai atau tidak

```
int isBalanced(char exp[]) {
    int i = 0;
    Stack* stack = NULL;
    while (exp[i]) {
        if (exp[i] == '{' || exp[i] == '(' || exp[i] == '[')
            push(&stack, exp[i]);
        if (exp[i] == '}' || exp[i] == ')' || exp[i] == ']') {
            if (stack == NULL) return 0;
            else if (!isMatchingPair(pop(&stack), exp[i])) return 0;
        }
        i++;
    }
    if (stack == NULL) return 1;
    else return 0;
```

baris ini berfungsi untuk memeriksa apakah urutan tanda kurung dalam string `exp` seimbang tau tidak

`int isBalanced(char exp[])` baris ini mengembil sebuah array karekter `exp` sebagai input dan mengambil nilai 1 jika tanda kurung dalam `exp` seimbang, dan 0 jika tidak

```
int main() {
    char exp[100];
    printf("Masukkan urutan tanda kurung: ");
    scanf("%s", exp);
    if (isBalanced(exp))
        printf("YES\n");
    else
        printf("NO\n");
    return 0;
```

`int main` berfungsi sebagai fungsi utama dari program

`char exp[100]` mendefenisikan array yang akan menyimpan urutan tanda kurung

`scanf("%s", exp);` membaca inputan dari pengguna dan menyimpannya dalam array `exp`

`if (isBalanced(exp)` berfungsi untuk memanggil fungsi `isBalanced` untuk memeriksa keseimbangaan tanda kurung dan menampilkan hasil

3. OUTPUT

```
PS D:\pemograman dan structur data (semester 2)\output> & .\'stuck tanda kurung.exe'
Masukkan urutan tanda kurung: {([])}
YES
PS D:\pemograman dan structur data (semester 2)\output> & .\'stuck tanda kurung.exe'
Masukkan urutan tanda kurung: ([)][]
NO
PS D:\pemograman dan structur data (semester 2)\output> & .\'stuck tanda kurung.exe'
Masukkan urutan tanda kurung: {{([])}}
YES
PS D:\pemograman dan structur data (semester 2)\output> & .\'stuck tanda kurung.exe'
Masukkan urutan tanda kurung: {{))]{(
NO
PS D:\pemograman dan structur data (semester 2)\output>
```