NAMA :ELDHIEN ANGGI DERMAWAN RAMBE
NIM : 1203230109

TUGAS OTH

1. SOURCODE

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

struct link {
    char data;
    struct link *next;
};

void linked(struct link **head, char data) {
    struct link *newlink = (struct link *)malloc(sizeof(struct link));
    if (newlink == NULL) {
        printf("Memory allocation failed!\n");
        exit(EXIT_FAILURE);
    }
    newlink->data = data;
    newlink->next = NULL;

    if (*head == NULL) {
        *head = newlink;
        return;
    }

    struct link *lastlink = *head;
    while (lastlink->next != NULL) {
        lastlink = lastlink->next;
    }
    lastlink->next = newlink;
}

void printlinks(struct link *head) {
    struct link *current = head;
    while (current != NULL) {
        printf("%c", current->data);
        current = current->next;
    }
}

void freelinks(struct link **head) {
    struct link *current = *head;
    struct link *next;
    while (current != NULL) {
        next = current->next;
```

```
        free(current);
        current = next;
    }
    *head = NULL;
}

int main() {
    struct link *head = NULL;

    // penambahan karakter di link
    linked(&head, 'I');
    linked(&head, 'N');
    linked(&head, 'F');
    linked(&head, 'O');
    linked(&head, 'R');
    linked(&head, 'M');
    linked(&head, 'A');
    linked(&head, 'T');
    linked(&head, 'I');
    linked(&head, 'K');
    linked(&head, 'A');

    printlinks(head);

    freelinks(&head);

    return 0;
}
```

2. PENJELASAN

```
#include <stdio.h>

#include <stdlib.h>
```

Pada baris ini sebagai header file pada Bahasa c, `<stdio.h>` berfungsi untuk menerima input dan out put, `<stdlib.h>` berfungsi untuk alokasi memori pada pemograman ini

```
struct link {
    char data;
    struct link *next;
```

baris ini mendefenisikan struktur `link` setiap node dalam linked akan memiliki sebuah karakter `data` dan pointer ke node `next`

```
void linked(struct link **head, char data) {
```

`linked` dedefinisikan dengan mengambil parameter pointer ke pointer `**head` dan karakter `data`

```
struct link *newlink = (struct link *)malloc(sizeof(struct link));
    if (newlink == NULL) {
        printf("Memory allocation failed!\n");
        exit(EXIT_FAILURE);
    }
```

Pada baris ini berfungsi untuk alokasi memori untuk node baru menggunakan `malloc` jika gagal (karena kekurangan memori) program akan akan mencetak kessalahan dan akan langsung keluar

```
newlink->data = data;
    newlink->next = NULL;
```

jika alokasih berhasil, karakter `data` disimpan didalam node baru dan pointer `next` di aatur menjadi null

```
if (*head == NULL) {
        *head = newlink;
        return;
    }
```

Pada baris ini akan mencek `(*head` apakah `NULL` jika iya maka akan menunkukkan linked list masih kosong, pointer `*head` akan di arahka ke node baru

```
 struct link *lastlink = *head;
    while (lastlink->next != NULL) {
        lastlink = lastlink->next;
    }
    lastlink->next = newlink;
}
```

Jika linked list tidak kosong, maka program akan mencari node terakhir dalam node linked list dan menambahkan node baru di belakang

```
void printlinks(struct link *head) {
    struct link *current = head;
    while (current != NULL) {
        printf("%c", current->data);
        current = current->next;
    }
}
```

Pada baris ini mendefenisikan fungsi `printlinks` untuk mencetak isi linked list. Fungsi ini menerima pointer `*head` ke linked list dan secara berurutan mencetak karakter dari node

```
void freelinks(struct link **head) {
    struct link *current = *head;
    struct link *next;
    while (current != NULL) {
        next = current->next;
        free(current);
        current = next;
    }
    *head = NULL;
```

Cpada baris ini menddefenisikan fungsi `freelinks` untuk membebaskan memori yang di alokasikan untuk node dalam linked list, fungsi ini menerima ppointer ke pointer `*head` dari linked list

```
int main() {
```

awal mula pemograman

```
struct link *head = NULL;
```

membuat pointer `*head` menjadi `NULL` atau kosong agar bisa di tambahkan

```
    linked(&head, 'I');
    linked(&head, 'N');
    linked(&head, 'F');
    linked(&head, 'O');
    linked(&head, 'R');
    linked(&head, 'M');
    linked(&head, 'A');
    linked(&head, 'T');
    linked(&head, 'I');
    linked(&head, 'K');
    linked(&head, 'A');
```

ini yang akan di tambah kan ke dalam pointer `head`

```
printlinks(head);

    freelinks(&head);
```

`printlinks(head)` berfungsi untuk memanggil fungsi `printlinks` untuk mencetak linked list

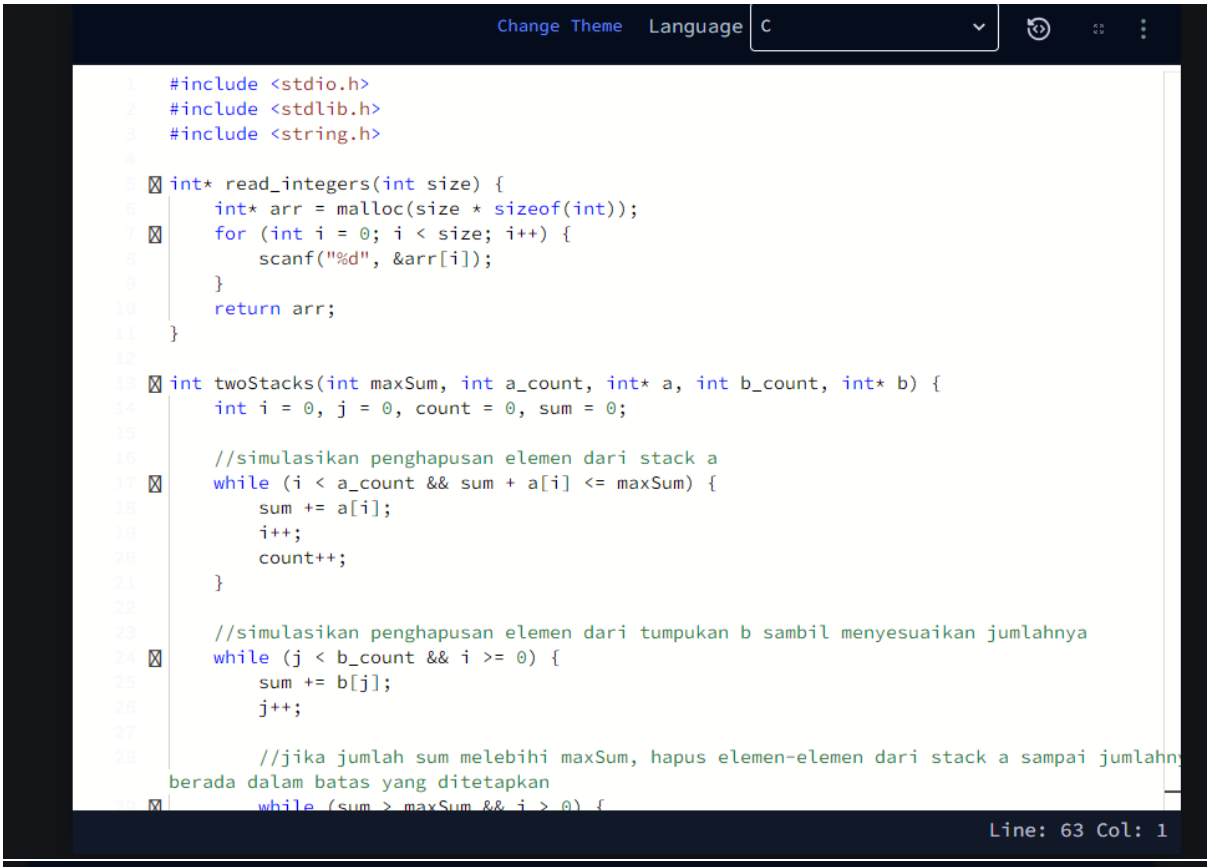
`freelinks(&head);` berfungsi untuk memanggil `freelinks` untuk membebaskan memori linked list

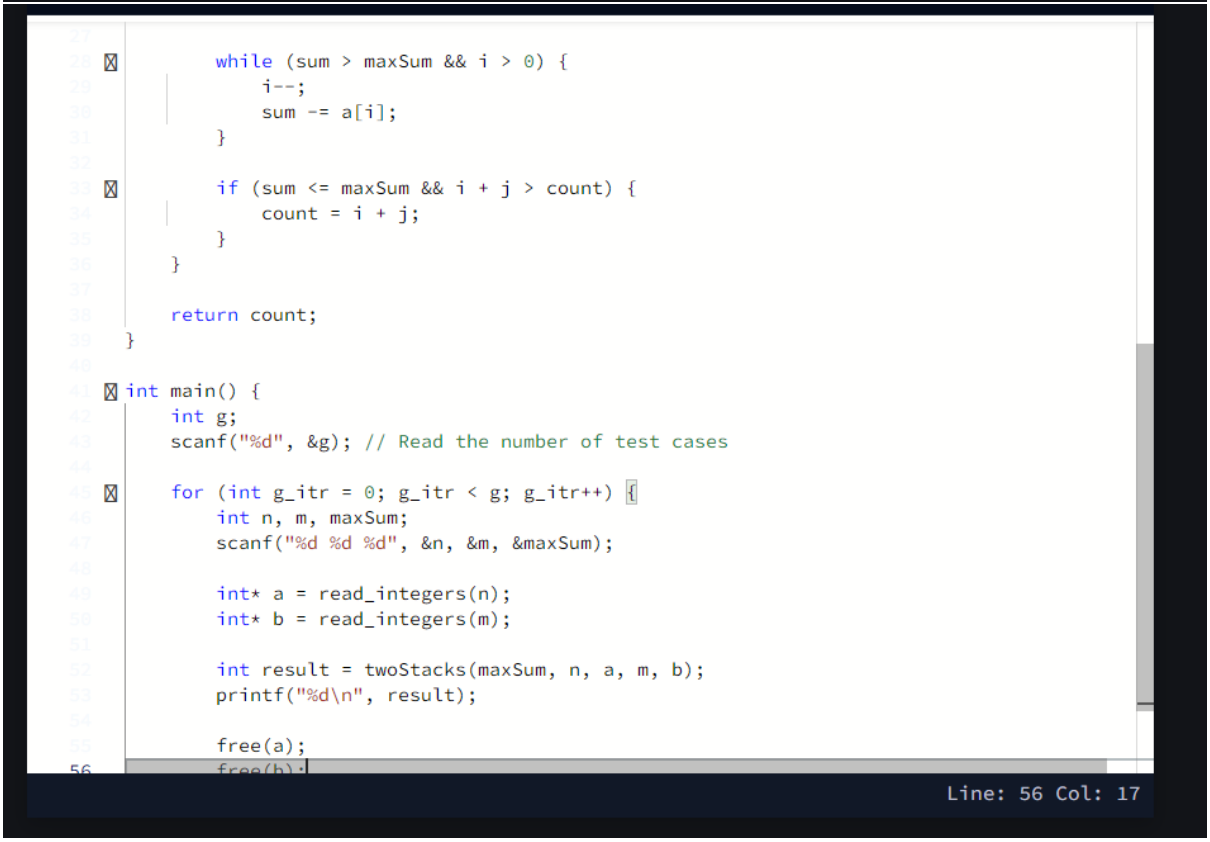
3. OUTPUT

```
PS D:\pemograman dan structur data (semester 2)> cd 'd:\pemograman dan structur data (semester 2)\output'
PS D:\pemograman dan structur data (semester 2)\output> & .\'tugas OTH.exe'
INFORMATIKA
PS D:\pemograman dan structur data (semester 2)\output>
```

TUGAS HACKER RANK

1. SOURCODE





2. PENJELASAN

```
 #include <stdio.h>

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

Pada baris ini sebagai header file pada Bahasa c, `<stdio.h>` berfungsi untuk menerima input dan out put, `<stdlib.h>` berfungsi untuk alokasi memori pada pemograman ini, dan `<string.h>` berfungsi sebagai pengelola string yang ada pada program ini\

```
int* read_integers(int size) {
    int* arr = malloc(size * sizeof(int));
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    return arr;
```

pada baris ini mendefenisikan fungsi `read_integers` yang membaca ukuran integer dari pengguna dan mengembalikan pointer ke array integer yang di baca `malloc` berfungsi alokasi memori

```
nt twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
    int i = 0, j = 0, count = 0, sum = 0;
```

Mendefinisikan fungsi `twoStacks` yang mengambil sebagai parameter `maxSum` (jumlah maksimum yang diizinkan), jumlah elemen dalam stack a `a_count`, array stack a jumlah elemen dalam stack b, dan array stack b

```

int main() {
    int g;
    scanf("%d", &g);

    for (int g_itr = 0; g_itr < g; g_itr++) {
        int n, m, maxSum;
        scanf("%d %d %d", &n, &m, &maxSum);

        int* a = read_integers(n);
        int* b = read_integers(m);

        int result = twoStacks(maxSum, n, a, m, b);
        printf("%d\n", result);

        free(a);
        free(b);
    }

    return 0;
```

`int main` awal mula program
`scanf("%d", &g);` berfungsi untuk membaca inputan pengguna, untuk setiap kasus uji, program akan melakukan loop sebanyak yang di masukkan pernggguna

`int n, m, maxSum` berfungsi deklarasi untuk menyimpan jumlah elemen dalam tumpukan a
`int* a = read_integers(n)` program akan membaca elemen elemen tumpukan a dari inputan [engguna dan menyimpan dalam array a
`int* b = read_integers(m)` program akan membaca elemen elemen tumpukan b dari inputan [engguna dan menyimpan dalam array b
`int result = twoStacks(maxSum, n, a, m, b` program akan memanggil `twoStacks` untuk menghitung jumlah maximum elemen yang dapat di ambil dari 2 tumpukan dan di simpan dalam `result`

3. OUTPUT