

OTH STACK**Source Code 1**

```
#include <stdio.h>

typedef struct node{ // Mendeklarasi tipe data struct 'Node' yang merupakan
sebuah struct
    char* alphabet; // Pointer ke karakter
    struct node* link; // Pointer ke node berikutnya dalam linked list
} node;

int main(){
    node l1, l2, l3, l4, l5, l6, l7, l8, l9; // Mendefinikan variabel bertipe
node yang digunakan untuk menyimpan huruf dan mengaitkannya dalam linked list

// Menginisialisasi huruf kedalam char alphabet. Setiap variabel
merepresentasikan satu huruf dan memiliki pointer 'link' yang diinisialisasi
menjadi 'NULL' dan pointer 'alphabet' yang menunjuk ke string yang berisi huruf
tersebut
    l1.link = NULL;
    l1.alphabet = "F";

    l2.link = NULL;
    l2.alphabet = "M";

    l3.link = NULL;
    l3.alphabet = "A";

    l4.link = NULL;
    l4.alphabet = "I";

    l5.link = NULL;
    l5.alphabet = "K";

    l6.link = NULL;
    l6.alphabet = "T";

    l7.link = NULL;
    l7.alphabet = "N";
```

```

18.link = NULL;
18.alphabet = "O";

19.link = NULL;
19.alphabet = "R";

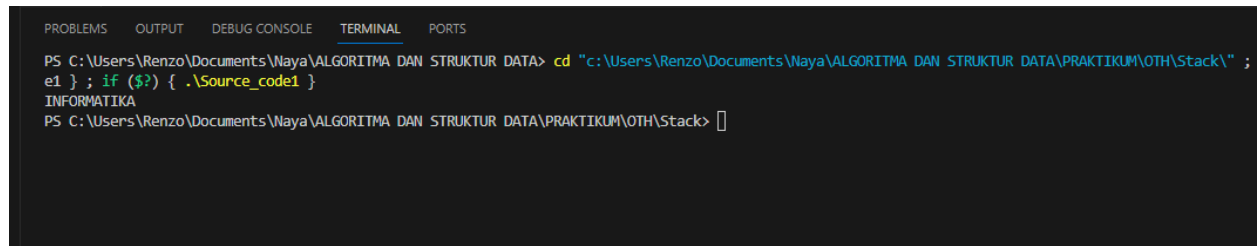
// Mengaitkan(linked) setiap node dalam linked list dimaulai dari huruf 'N'
17.link = &l1; // Mengaitkan huruf 'N' dengan 'F'
11.link = &l8; // Mengaitkan huruf 'F' dengan 'O'
18.link = &l2; // Mengaitkan huruf 'O' dengan 'M'
12.link = &l5; // Mengaitkan huruf 'M' dengan 'K'
15.link = &l3; // Mengaitkan huruf 'K' dengan 'A'
13.link = &l6; // Mengaitkan huruf 'A' dengan 'T'
16.link = &l9; // Mengaitkan huruf 'T' dengan 'R'
19.link = &l4; // Mengaitkan huruf 'R' dengan 'I'
14.link = &l7; // Mengaitkan huruf 'I' dengan 'N'

// Menampilkan setiap huruf yang di panggil sesuai yang terkait dalam linked list
dengan menggunakan pointer
printf("%s", 13.link -> link -> link -> alphabet); // Print huruf 'I'
printf("%s", 13.link -> link -> link -> link -> alphabet); // Print huruf 'N'
printf("%s", 13.link -> link -> link -> link -> link -> alphabet); // Print
huruf 'F'
printf("%s", 13.link -> link -> link -> link -> link -> link -> alphabet); //
Print huruf 'O'
printf("%s", 13.link -> link -> alphabet); // Print huruf 'R'
printf("%s", 13.link -> link -> link -> link -> link -> link -> link ->
alphabet); // Print huruf 'M'
printf("%s", 13.link -> link -> link -> link -> link -> link -> link -> link ->
link -> alphabet); // Print huruf 'A'
printf("%s", 13.link -> alphabet); // Print huruf 'T'
printf("%s", 13.link -> link -> link -> alphabet); // Print huruf 'I'
printf("%s", 13.link -> link -> link -> link -> link -> link -> link -> link ->
alphabet); // Print huruf 'K'
printf("%s", 13.link -> link -> link -> link -> link -> link -> link -> link ->
link -> alphabet); // Print huruf 'A'

return 0; // Mengakhiri eksekusi program dan memberi nilai kembali ke sistem
operasi
}

```

Output :



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\Renzo\Documents\Naya\ALGORITMA DAN STRUKTUR DATA> cd "c:\Users\Renzo\Documents\Naya\ALGORITMA DAN STRUKTUR DATA\PRAKTIKUM\OTH\Stack\" ;
e1 } ; if ($?) { .\Source_code1 }
INFORMATIKA
PS C:\Users\Renzo\Documents\Naya\ALGORITMA DAN STRUKTUR DATA\PRAKTIKUM\OTH\Stack> █
```

Source Code 2

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

char** pisahkanString(char*); // Deklarasi fungsi ubahKeInteger yang
mengembalikan nilai integer dan menerima parameter string sebagai input.
Namun, fungsi ini tidak diimplementasikan dalam kode yang diberikan
int ubahKeInteger(char*); // Deklarasi fungsi ubahKeInteger yang mengembalikan
nilai integer dan menerima parameter string sebagai input.
int duaTumpukan(int, int, int*, int, int*); // Deklarasi fungsi duaTumpukan yang
mengembalikan nilai integer dan menerima beberapa parameter, yaitu nilai
maksimum, jumlah elemen tumpukan A, array tumpukan A, jumlah elemen tumpukan B,
dan array tumpukan B

int main() // Fungsi utama program yang akan di jalankan
{
    int g; // Deklarasi variabel g dengan tipe data integer
    scanf("%d", &g); // Untuk mengambil input integer dari pengguna dan
    menyimpannya di variabel g

    for (int g_itr = 0; g_itr < g; g_itr++) { // Melakukan iterasi sebanyak nilai
    g dan menggunakan variabel g_itr sebagai penghitung iterasi
        int n, m, maxSum; // Deklarasi variabel untuk menyimpan input jumlah
        elemen A, B, dan nilai maksimum yang diizinkan
        scanf("%d %d %d", &n, &m, &maxSum); // Untuk mengambil input integer dari
        pengguna dan menyimpannya di variabel n,m,dan maxSum

        int* a = malloc(n * sizeof(int)); // Mengalokasikan memori dinamis untuk
        array a dengan ukuran n * sizeof(int)
        for (int i = 0; i < n; i++) {
            scanf("%d", &a[i]); // Menggunakan scanf untuk mengambil input
            integer sebanyak n kali dan menyimpannya di dalam array a
        }

        int* b = malloc(m * sizeof(int)); // Mengalokasikan memori dinamis untuk
        array b dengan ukuran m * sizeof(int)
        for (int i = 0; i < m; i++) {
            scanf("%d", &b[i]); // Menggunakan scanf untuk mengambil input
            integer sebanyak m kali dan menyimpannya di dalam array b
        }
    }
```

```

    }

    int result = duaTumpukan(maxSum, n, a, m, b); // Memanggil fungsi
duaTumpukan dengan argumen nilai maksimum, jumlah elemen tumpukan

    printf("%d\n", result); // Mencetak hasil dari fungsi duaTumpukan

    free(a); // Membebaskan memori yang dialokasikan untuk array a
    free(b); // Membebaskan memori yang dialokasikan untuk array b
}

    return 0; // Mengembalikan nilai 0 untuk menunjukkan bahwa program berakhir
}
int duaTumpukan(int maxSum, int a_jumlah, int* a, int b_jumlah, int* b) {
    // eklarasi fungsi duaTumpukan yang mengambil beberapa parameter
maxSum (nilai maksimum yang diizinkan), a_jumlah (jumlah elemen dalam tumpukan
A), a (array tumpukan A), b_jumlah (jumlah elemen dalam tumpukan B), dan b (array
tumpukan B). Fungsi ini mengembalikan nilai integer
    int jumlah = 0; // Deklarasi variabel lokal jumlah yang akan digunakan untuk
menghitung jumlah elemen
    int jumlahSementara = 0; // Deklarasi variabel lokal jumlahSementara yang
akan digunakan untuk menyimpan total nilai elemen yang dipilih saat ini dari
kedua tumpukan
    int idx_a = 0, idx_b = 0; // Deklarasi variabel lokal idx_a dan idx_b yang
akan digunakan sebagai indeks untuk mengakses elemen-elemen dari tumpukan A dan
B.

    while (idx_a < a_jumlah && jumlahSementara + a[idx_a] <= maxSum) { // Memulai
loop while yang akan berjalan selama indeks idx_a masih dalam rentang jumlah
elemen tumpukan A dan jumlahSementara ditambah dengan nilai elemen tumpukan A
pada indeks idx_a masih kurang dari atau sama dengan maxSum
        jumlahSementara += a[idx_a]; // Menambahkan nilai elemen tumpukan A pada
indeks idx_a ke dalam jumlahSementara
        idx_a++; // Menaikkan nilai indeks idx_a untuk memeriksa elemen
selanjutnya dari tumpukan A
        jumlah++; // Menaikkan nilai jumlah untuk menghitung jumlah elemen yang
telah dipilih dari kedua tumpukan
    }

    int maksJumlahElemen = jumlah; // Menginisialisasi variabel maksJumlahElemen
dengan nilai jumlah. Variabel ini akan digunakan untuk menyimpan jumlah maksimum
elemen

```

```

    while (idx_b < b_jumlah && idx_a >= 0) { // Memulai loop while kedua yang
akan berjalan selama indeks idx_b masih dalam rentang jumlah elemen tumpukan B
dan indeks idx_a tidak kurang dari 0
        jumlahSementara += b[idx_b]; // Menambahkan nilai elemen tumpukan B pada
indeks idx_b ke dalam jumlahSementara
        idx_b++; // Menaikkan nilai indeks idx_b untuk memeriksa elemen
selanjutnya dari tumpukan B
        jumlah++; // Menaikkan nilai jumlah untuk menghitung jumlah elemen yang
telah dipilih dari kedua tumpukan

        while (jumlahSementara > maxSum && idx_a > 0) { // Memulai loop while
bersarang yang akan berjalan selama jumlahSementara lebih besar dari maxSum dan
indeks idx_a masih lebih dari 0
            idx_a--; // Mengurangi nilai indeks idx_a untuk memeriksa elemen
sebelumnya dari tumpukan A
            jumlahSementara -= a[idx_a]; // Mengurangi nilai elemen tumpukan A
pada indeks idx_a dari jumlahSementara
            jumlah--; // Mengurangi nilai jumlah untuk mencerminkan pengurangan
satu elemen dari tumpukan A
        }

        if (jumlahSementara <= maxSum && jumlah > maksJumlahElemen) { //
Memeriksa apakah jumlahSementara kurang dari atau sama dengan maxSum dan jumlah
lebih besar dari maksJumlahElemen
            maksJumlahElemen = jumlah; // jika kondisi di atas terpenuhi, maka
variabel maksJumlahElemen akan diperbarui dengan nilai jumlah
        }
    }

    return maksJumlahElemen; // Mengembalikan nilai maksJumlahElemen sebagai
hasil dari fungsi duaTumpukan
}

```

Output :

✓ **Sample Test case 0**

Input (stdin)

[Download](#)

1	1
2	5 4 10
3	4 2 4 6 1
4	2 1 8 5

Your Output (stdout)

1	4
---	---

Expected Output

[Download](#)

1	4
---	---
