

LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

NAMA : AISYAH

NIM : 1203230015

KELAS : IF-03-03

1. Source code & penjelasan

```
1 #include <stdio.h> //berfungsi sbg standar input output
2 #include <stdlib.h> // berfungsi untuk mengalokasikan blok memori secara dinamis
3
4 //definisi struktur kim
5 typedef struct Kim {
6     char *alphabet; // berfungsi untuk menyimpan huruf pada batu
7     struct Kim *link; // Pointer ke node berikutnya
8 } Kim;
9
10 int main() { // untuk membuka atau memulai membuat program dan fungsi utama
11     // berfungsi deklarasi dan inisialisasi node-node batu
12     Kim l1 = {.link = NULL, .alphabet = "F"};
13     Kim l2 = {.link = NULL, .alphabet = "M"};
14     Kim l3 = {.link = NULL, .alphabet = "A"};
15     Kim l4 = {.link = NULL, .alphabet = "I"};
16     Kim l5 = {.link = NULL, .alphabet = "K"};
17     Kim l6 = {.link = NULL, .alphabet = "T"};
18     Kim l7 = {.link = NULL, .alphabet = "N"};
19     Kim l8 = {.link = NULL, .alphabet = "O"};
20     Kim l9 = {.link = NULL, .alphabet = "R"};
21
22     // berfungsi untuk mengatur koneksi antar batu sesuai dengan urutan yang sudah diberikan
23     l3.link = &l4;
24     l4.link = &l5;
25     l5.link = &l6;
26     l6.link = &l7;
27     l7.link = &l8;
28     l8.link = &l9;
29     l9.link = &l1;
30     l1.link = &l2;
31
32     // berfungsi untuk mengakses huruf pada batu menggunakan l3 sebagai titik awal
33     printf("%s", l3.link->alphabet); // Output: I
34     printf("%s", l3.link->link->link->link->alphabet); // Output: N
35     printf("%s", l3.link->link->link->link->link->link->link->alphabet); // Output: F
36     printf("%s", l3.link->link->link->link->link->alphabet); // Output: O
37     printf("%s", l3.link->link->link->link->link->link->link->alphabet); // Output: R
38     printf("%s", l3.link->link->link->link->link->link->link->link->link->link->link->alphabet); // Output: M
39     printf("%s", l3.alphabet); // Output: A
40     printf("%s", l3.link->link->link->link->link->alphabet); // Output: T
41     printf("%s", l3.link->alphabet); // Output: I
42     printf("%s", l3.link->link->alphabet); //Output : K
43     printf("%s", l3.alphabet); // Output: A
44
45     return 0; // mengembalikan nilai 0 untuk menandakan program berakhir dengan sukses
46 }
```

2. Output

```
} ; if ($?) { .\Informatika }
INFORMATIKA
PS C:\Users\istiyono\SEMESTER 2.C>
```

3. Source code & penjelasan




```
1 #include <stdio.h> //berfungsi untuk standar input output
2
3 int twoStacks(int maxSum, int a[], int n, int b[], int m) { //berfungsi untuk menghitung jumlah maksimum elemen yg dapat diambil dari kedua
4                                     //tumpukan tanpa melebihi 'maxSum'
5     int sum = 0, count = 0, temp = 0, i = 0, j = 0; //deklarasi variabel yg digunakan dalam perhitungan
6
7     while (i < n && sum + a[i] <= maxSum) { //melakukan iterasi melalui tumpukan pertama dan tidak melebihi nilai maksimum
8         sum += a[i++]; //pada setiap iterasi perulangan ditingkatkan sehingga iterasi berikutnya akan diakses tumpukan selanjutnya
9     }
10    count = i; //digunakan sbg hasil akhir untuk menghitung jumlah maksimum elemen yg dapat diambil dari tumpukan pertama
11
12    while (j < m && i >= 0) { //melakukan iterasi selama msh tersisa elemen dalam tumpukan kedua dan msh ada elemen yg diambil dari tumpukan pertama
13        sum += b[j++];
14        while (sum > maxSum && i > 0) {
15            sum -= a[--i];
16        }
17        if (sum <= maxSum && i + j > count) { //jika melebihi 'maxSum' elemen" dari tumpukan pertama akan dihapus satu per satu
18            count = i + j; //hingga 'sum' kembali ke nilai yg dapat diterima
19        }
20    }
21    return count; //mengembalikan nilai yg merupakan jumlah max elemen yg dapat diambil dari kedua tumpukan tanpa melebihi 'maxSum'
22 }
23
24 int main() { // untuk membuka atau memulai membuat program dan fungsi utama
25     int g; //deklarasi variabel yg digunakan untuk menyimpan jumlah kasus yg di uji
26     scanf("%d", &g); //berfungsi untuk membaca inputan dari user
27     while (g--) { //loop yg akan terus berjalan dan digunakan untuk mengurangi nilai 'g' setiap kali loop dijalankan
28         int n, m, maxSum; //deklarasi variabel yg akan digunakan untuk menyimpan jumlah elemen dalam tumpukan pertama
29         scanf("%d%d%d", &n, &m, &maxSum); //berfungsi untuk membaca 3 inputan dari pengguna
30         int a[n], b[m]; //deklarasi variabel untuk menyimpan elemen" dari tumpukan pertama
31         for (int i = 0; i < n; i++) { //loop yg digunakan untuk membaca elemen" dari tumpukan pertama
32             scanf("%d", &a[i]); //berfungsi untuk membaca inputan dari user di elemen tumpukan pertama
33         }
34         for (int i = 0; i < m; i++) { //loop yg digunakan untuk membaca elemen" dari tumpukan kedua
35             scanf("%d", &b[i]); //berfungsi untuk membaca inputan dari user di elemen tumpukan kedua
36         }
37         printf("%d\n", twoStacks(maxSum, a, n, b, m)); //berfungsi untuk mencetak 'twoStacks' yg hasilnya dikembalikan oleh fungsi 'twoStacks'
38     }
39     return 0; // berfungsi untuk mengakhiri fungsi dan program
40 }
```

4. Output

```
1
5 4 11
4 5 2 1 1
3 1 1 2
5
PS C:\Users\istiyono\SEMESTER 2.C\output\PRAKTIKUM>
```

Congratulations

You solved this challenge. Would you like to challenge your friends?



Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Input (stdin)

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

Expected Output

1	4
---	---