LAPORAN PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA

MATERI: STRUCT DAN STACK

Oleh:

Shinta Putri Nirmala

NIM: 1203230052

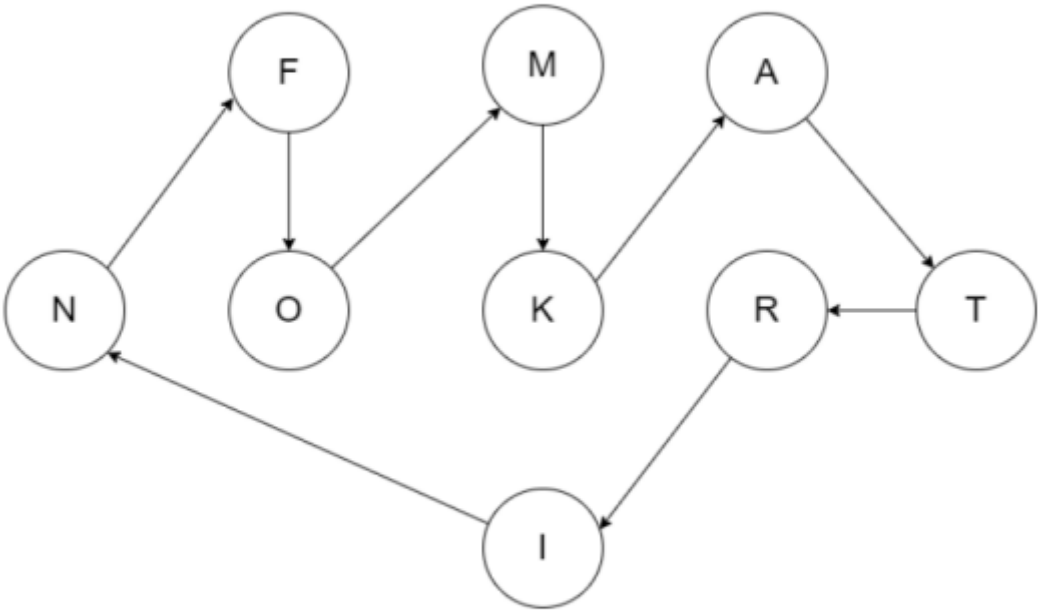
Kelas: IF 03-01

PROGRAM STUDI INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM SURABAYA
TAHUN 2024

SOAL PERTAMA

Asisten Sherlock Holmes

Tugasmu kali ini adalah memecahkan teka-teki berantai yang tersusun dalam serangkaian batu yang tersebar di dalam hutan. Petunjuk-petunjuk untuk menyelesaikan teka-teki ini tersimpan dalam urutan batu-batu yang saling terhubung dengan tanda panah, membentuk pola yang harus diikuti untuk menemukan solusi. Berikut merupakan rangkaian batu yang telah kamu amati.



Pada setiap batu tersebut saling terhubung dengan tanda panah yang dimulai dari batu dengan huruf **“N”** dan seterusnya mengikuti arah panah. Ternyata, jika kita mengikuti arah panah dari batu dengan huruf **"N"**, dan menggabungkan huruf-huruf pada batu-batu tersebut, kita akan membentuk sebuah kata yang sangat penting: "INFORMATIKA". Tapi tunggu, Sherlock Holmes memberimu sebuah catatan dalam bentuk kode yang akan membantumu menyelesaikan teka-teki ini. Kode tersebut menggunakan konsep **Self Referential Structures** untuk merepresentasikan hubungan antar batu-batu.

```
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";

l8.link = NULL;
l8.alphabet = "O";

l9.link = NULL;
l9.alphabet = "R";
```

Tugasmu sekarang adalah merangkai huruf tersebut hingga membentuk suatu kata yang telah ditentukan dengan menggunakan **Self Referential Structures** dan ketentuan sebagai berikut.

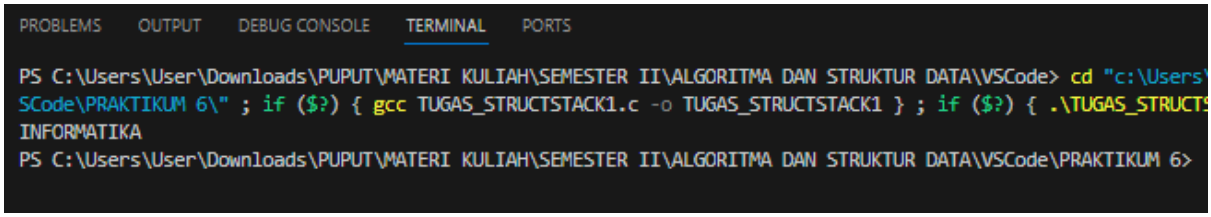
1. **Wajib** menggunakan potongan kode yang ada pada tabel diatas untuk inisialisasi (tidak ada penambahan / pengurangan ketika inisialisasi).
2. Pastikan ketika membuat linking / koneksi **harus sesuai** dengan gambar diatas (urutan dimulai dari huruf **N** dan mengikuti arah panah).
3. Lakukan semua akses data dengan menggunakan **l3 sebagai starting point**.

SOURCE CODE

```c
#include <stdio.h> // Mengimpor library standar input-output
#include <stdlib.h> // Mengimpor library standar untuk fungsi-fungsi umum

struct Node { // Deklarasi struktur Node
    char* alphabet; // Mendefinisikan pointer ke karakter untuk menyimpan huruf
    struct Node* link; // Mendefinisikan pointer ke Node berikutnya
};

int main() { // Fungsi utama program
    // Deklarasi node-node
    struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9; // Mendeklarasikan node-node
    // Deklarasi pointer ke Node
    struct Node *l3ptr;

    // Inisialisasi node-node dengan menggunakan potongan kode
    l1.link = NULL; //Mengatur link dari l1 ke NULL
    l1.alphabet = "F"; //Mengatur nilai alphabet dari l1 menjadi "F"
    l2.link = NULL; //Mengatur link dari l2 ke NULL
    l2.alphabet = "M"; //Mengatur nilai alphabet dari l2 menjadi "M"
    l3.link = NULL; //Mengatur link dari l3 ke NULL
    l3.alphabet = "A"; //Mengatur nilai alphabet dari l3 menjadi "A"
    l4.link = NULL; //Mengatur link dari l4 ke NULL
    l4.alphabet = "I"; //Mengatur nilai alphabet dari l4 menjadi "I"
    l5.link = NULL; //Mengatur link dari l5 ke NULL
    l5.alphabet = "K"; //Mengatur nilai alphabet dari l5 menjadi "K"
    l6.link = NULL; //Mengatur link dari l6 ke NULL
    l6.alphabet = "T"; //Mengatur nilai alphabet dari l6 menjadi "T"
    l7.link = NULL; //Mengatur link dari l7 ke NULL
    l7.alphabet = "N"; //Mengatur nilai alphabet dari l7 menjadi "N"
    l8.link = NULL; //Mengatur link dari l8 ke NULL
    l8.alphabet = "O"; //Mengatur nilai alphabet dari l8 menjadi "O"
    l9.link = NULL; //Mengatur link dari l9 ke NULL
    l9.alphabet = "R"; //Mengatur nilai alphabet dari l9 menjadi "R"

    // Mengatur koneksi antar node sesuai dengan urutan yang diinginkan
    l7.link = &l1; // N -> F
    l1.link = &l8; // F -> O
    l8.link = &l2; // O -> M
    l2.link = &l5; // M -> K
    l5.link = &l3; // K -> A
    l3.link = &l6; // A -> T
    l6.link = &l9; // T -> R
    l9.link = &l4; // R -> I
    l4.link = &l7; // I -> N
```

```
    // Starting point
    l3ptr = &l3;

    // Menampilkan huruf sesuai dengan jalur koneksi node
    printf("%s", l3.link->link->link->alphabet); // Menampilkan huruf I dengan
jalur koneksi node l3.link ke l4.link

    printf("%s", l3.link->link->link->link->alphabet); // Menampilkan huruf N
dengan jalur koneksi node l3.link ke l7.link

    printf("%s", l3.link->link->link->link->link->alphabet); // Menampilkan
huruf F dengan jalur koneksi node l3.link ke l1.link

    printf("%s", l3.link->link->link->link->link->link->alphabet); //
Menampilkan huruf O dengan jalur koneksi node l3.link ke l8.link

    printf("%s", l3.link->link->alphabet); // Menampilkan huruf R dengan jalur
koneksi node l3.link ke l9.link

    printf("%s", l3.link->link->link->link->link->link->link->alphabet); //
Menampilkan huruf M dengan jalur koneksi node l3.link ke l2.link

    printf("%s", l3.alphabet); // Menampilkan huruf A dari jalur koneksi node
l3.link

    printf("%s", l3.link->alphabet); // Menampilkan huruf T dengan jalur
koneksi node l3.link ke l6.link

    printf("%s", l3.link->link->link->alphabet); // Menampilkan huruf I dengan
jalur koneksi node l3.link ke l4.link

    printf("%s", l3.link->link->link->link->link->link->link->link->alphabet);
// Menampilkan huruf K dengan jalur koneksi node l3.link ke l5.link

    printf("%s", l3.alphabet); // Menampilkan huruf A dari jalur koneksi node
l3.link

    return 0; //Mengakhiri fungsi main() dengan mengembalikan nilai 0,
menandakan bahwa program berakhir tanpa kesalahan
}
```

OUTPUT


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\User\Downloads\PUPUT\MATERI KULIAH\SEMESTER II\ALGORITMA DAN STRUKTUR DATA\VSCode> cd "c:\Users
SCode\PRAKTIKUM 6\" ; if ($?) { gcc TUGAS_STRUCTSTACK1.c -o TUGAS_STRUCTSTACK1 } ; if ($?) { .\TUGAS_STRUCT
INFORMATIKA
PS C:\Users\User\Downloads\PUPUT\MATERI KULIAH\SEMESTER II\ALGORITMA DAN STRUKTUR DATA\VSCode\PRAKTIKUM 6>

SOAL 2

Selesaikan soal pada link di bawah ini!

https://www.hackerrank.com/challenges/game-of-two-stacks/problem

Tambahkan **visualisasi** alur penyelesaiannya (per langkah) dalam bentuk stack pada laporan apabila input diubah menjadi sebagai berikut.

1
5 4 11
4 5 2 1 1
3 1 1 2

Example

$a = [1, 2, 3, 4, 5]$

$b = [6, 7, 8, 9]$

The maximum number of values Nick can remove is 4. There are two sets of choices with this result.

1. Remove $1, 2, 3, 4$ from a with a sum of 10.
2. Remove $1, 2, 3$ from a and 6 from b with a sum of 12.

Function Description

Complete the twoStacks function in the editor below.

twoStacks has the following parameters: - int maxSum: the maximum allowed sum

- int a[n]: the first stack

- int b[m]: the second stack

Returns

- int: the maximum number of selections Nick can make

Input Format

The first line contains an integer, g (the number of games). The $3 \cdot g$ subsequent lines describe each game in the following format:

1. The first line contains three space-separated integers describing the respective values of n (the number of integers in stack a), m (the number of integers in stack b), and $maxSum$ (the number that the sum of the integers removed from the two stacks cannot exceed).
2. The second line contains n space-separated integers, the respective values of $a[i]$.
3. The third line contains m space-separated integers, the respective values of $b[i]$.

Constraints

- $1 \leq g \leq 50$
- $1 \leq n, m \leq 10^5$
- $0 \leq a[i], b[i] \leq 10^6$
- $1 \leq maxSum \leq 10^9$

Subtasks

- $1 \leq n, m, \leq 100$ for 50% of the maximum score.

Sample Input 0

```
1
5 4 10
4 2 4 6 1
2 1 8 5
```

Sample Output 0

```
4
```

Explanation 0

The two stacks initially look like this:

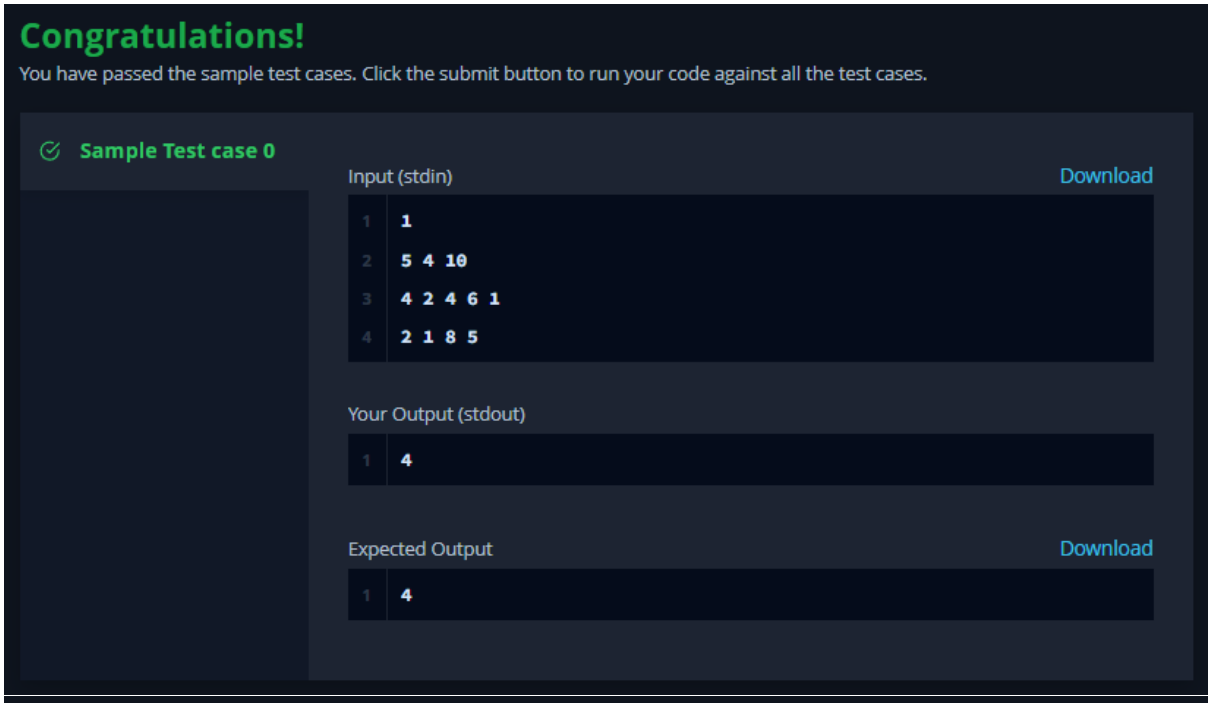
SOURCE CODE (FUNGSI twoStacks & main)

```
int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {

    int count = 0; //Mendeklarasikan variabel lokal count dan
menginisialisasinya dengan nilai 0. Variabel ini akan digunakan untuk
menghitung jumlah elemen yang diambil dari kedua tumpukan
    int temporarySum = 0; //Mendeklarasikan variabel lokal temporarySum dan
menginisialisasinya dengan nilai 0. Variabel ini akan digunakan untuk
menyimpan jumlah sementara elemen yang diambil dari tumpukan
    int idx_a = 0, idx_b = 0; //Mendeklarasikan dan menginisialisasikan dua
variabel lokal idx_a dan idx_b dengan nilai 0. Variabel ini digunakan sebagai
indeks untuk mengakses elemen dari tumpukan a dan b

    // Loop untuk mengambil bilangan bulat dari tumpukan pertama (a) sebanyak
mungkin
    while (idx_a < a_count && temporarySum + a[idx_a] <= maxSum) { //oop untuk
mengambil elemen dari tumpukan pertama (a) sebanyak mungkin selama jumlah
sementara elemen yang diambil tidak melebihi maxSum
        temporarySum += a[idx_a]; //Menambahkan nilai elemen yang diambil dari
tumpukan pertama ke temporarySum
        idx_a = idx_a + 1; //Meningkatkan indeks idx_a untuk mengambil elemen
berikutnya dari tumpukan pertama
        count = count + 1; //Menambahkan 1 ke jumlah total elemen yang telah
diambil
    }

    // Simpan jumlah maksimum elemen yang diambil dari tumpukan pertama
    int maxElementCount = count;

    // Loop untuk mengambil bilangan bulat dari tumpukan kedua (b) dan
memperbarui jumlah bilangan bulat yang diambil
    while (idx_b < b_count && idx_a >= 0) { //Mulai dari sini adalah loop
untuk mengambil elemen dari tumpukan kedua (b) dan memeriksa apakah jumlah
elemen yang diambil melebihi jumlah maksimum yang telah ditentukan sebelumnya
        temporarySum += b[idx_b]; //Menambahkan nilai elemen yang diambil dari
tumpukan kedua ke temporarySum
        idx_b = idx_b + 1; //Meningkatkan indeks idx_b untuk mengambil elemen
berikutnya dari tumpukan kedua
        count = count + 1; //Menambahkan 1 ke jumlah total elemen yang telah
diambil

        // Memastikan total jumlah bilangan bulat yang diambil tidak melebihi
maxSum
        while (temporarySum > maxSum && idx_a > 0) { //Memulai loop untuk
memastikan total jumlah elemen yang diambil tidak melebihi maxSum
            idx_a = idx_a - 1; //Mengurangi indeks idx_a untuk mengurangi
jumlah elemen dari tumpukan pertama
```

```
            temporarySum -= a[idx_a]; //Mengurangi nilai elemen yang diambil
dari temporarySum
            count = count - 1; //Mengurangi 1 dari jumlah total elemen yang
telah diambil
        }

        // Memeriksa apakah jumlah elemen saat ini melebihi jumlah elemen
maksimum sebelumnya
        if (temporarySum <= maxSum && count > maxElementCount) { //Memeriksa
apakah jumlah elemen yang diambil sekarang tidak melebihi maxSum dan jika iya,
apakah jumlah elemen saat ini lebih besar dari jumlah elemen maksimum yang
telah disimpan sebelumnya
            maxElementCount = count; //ika ya, maka jumlah elemen saat ini
menjadi jumlah maksimum yang baru
        }
    }

    return maxElementCount; //Mengembalikan jumlah maksimum elemen yang
diambil dari kedua tumpukan
}

int main() {
    int g; //Mendeklarasikan variabel g untuk menyimpan jumlah permainan
    scanf("%d", &g); //Meminta input dari pengguna untuk jumlah permainan

    for (int g_itr = 0; g_itr < g; g_itr++) { //Loop untuk setiap permainan.
        int n, m, maxSum; //Mendeklarasikan variabel untuk menyimpan jumlah
elemen dalam tumpukan a dan b, serta jumlah maksimum yang tidak boleh dilewati
        scanf("%d", &n); //Meminta input untuk jumlah elemen dalam tumpukan
pertama a
        scanf("%d",&m); //Meminta input untuk jumlah elemen dalam tumpukan
kedua b
        scanf("%d",&maxSum); //Meminta input untuk jumlah maksimum yang tidak
boleh dilewati

        int* a = malloc(n * sizeof(int)); //Mengalokasikan memori untuk
tumpukan a
        for (int i = 0; i < n; i++) { //Loop untuk mengisi elemen tumpukan a
        scanf("%d", &a[i]);   //Meminta input untuk setiap elemen tumpukan a
        }

        int* b = malloc(m * sizeof(int)); //Mengalokasikan memori untuk
tumpukan b
        for (int i = 0; i < m; i++) { //Loop untuk mengisi elemen tumpukan b
            scanf("%d", &b[i]); //Meminta input untuk setiap elemen tumpukan b
        }
```

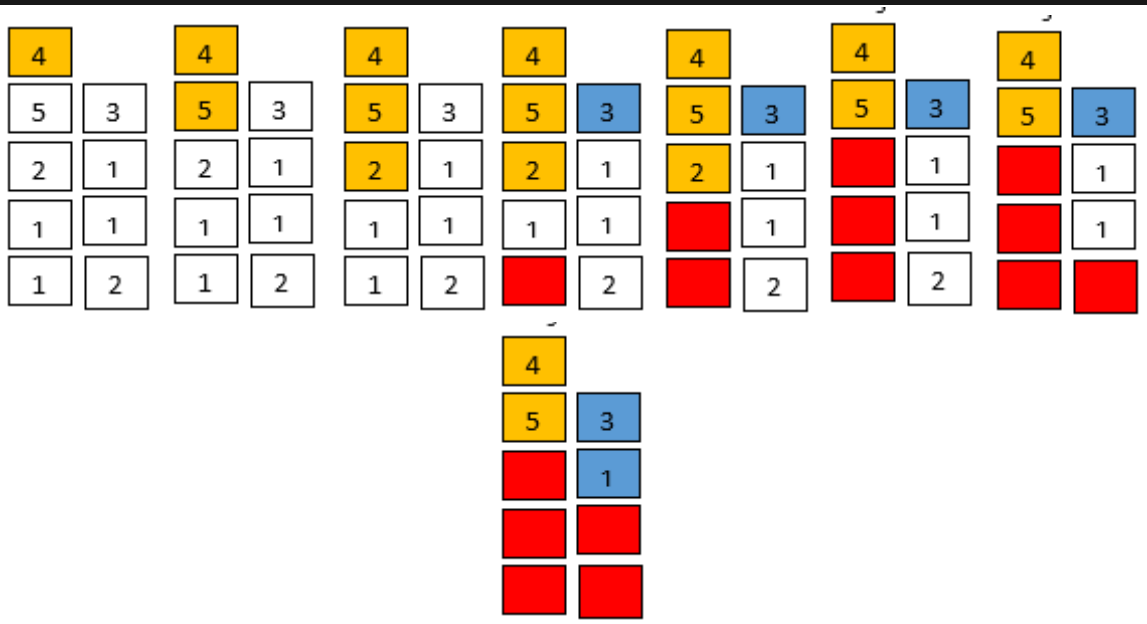
```
        int result = twoStacks(maxSum, n, a, m, b); //Memanggil fungsi
twoStacks untuk menghitung jumlah maksimum elemen yang dapat diambil
        printf("%d\n", result); //Menampilkan hasil jumlah maksimum elemen
yang dapat diambil

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

    return 0; //Mengembalikan nilai 0 untuk menunjukkan bahwa program berakhir
}
```

OUTPUT


Congratulations!
You have passed the sample test cases. Click the submit button to run your code against all the test cases.
Sample Test case 0
Input (stdin)
Download
1
5 4 10
4 2 4 6 1
2 1 8 5
Your Output (stdout)
4
Expected Output
Download
4


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)
Download
1
5 4 10
4 2 4 6 1
2 1 8 5
Expected Output
Download
4

```
PS C:\Users\User\Downloads\PUPUT\MATERI KULIAH\SEMESTER II\ALGORITMA DAN STRUKTUR DATA\VSCode> cd "c:\Users\User\
SCode\PRAKTIKUM 6\" ; if ($?) { gcc TUGAS_STRUCTSTAK2.c -o TUGAS_STRUCTSTAK2 } ; if ($?) { .\TUGAS_STRUCTSTAK2 }
1
5 4 11
4 5 2 1 1
3 1 1 2
5
PS C:\Users\User\Downloads\PUPUT\MATERI KULIAH\SEMESTER II\ALGORITMA DAN STRUKTUR DATA\VSCode\PRAKTIKUM 6>
```



Tumpukan a: [4, 5, 2, 1, 1]

Tumpukan b: [3, 1, 1, 2]

Mengambil [4], total sementara = 4

Mengambil [5], total sementara = 4 + 5 = 9

Mengambil [2], total sementara = 9 + 2 = 11 (mencapai maksimum)

Hapus [1] dari tumpukan a, total sementara = 11 - 1 = 10 (masih melebihi maksimum)

Hapus [1] dari tumpukan a, total sementara = 10 - 1 = 9 (masih melebihi maksimum)

Hapus [2] dari tumpukan a, total sementara = 9 - 2 = 7 (tidak melebihi maksimum)

Mengambil [3] dari tumpukan b, total sementara = 7 + 3 = 10 (masih tidak melebihi maksimum)

Mengambil [1] dari tumpukan b, total sementara = 10 + 1 = 11 (mencapai maksimum)