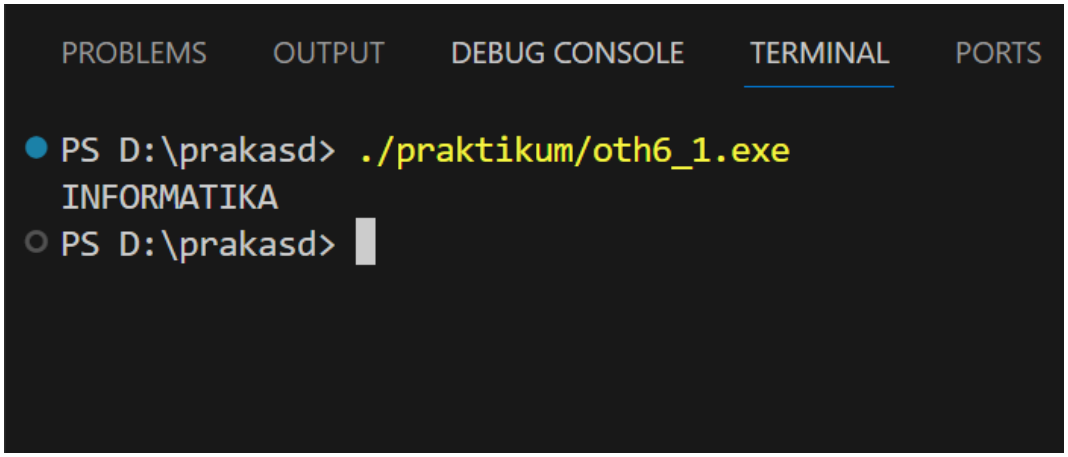
Nama : Ahmad Fitra Naufal

NIM : 1203230032

Kelas : IF 03-03

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

// Definisikan struktur Node untuk merepresentasikan simpul dalam linked list
struct Node {
    char* alphabet; // Menyimpan karakter huruf
    struct Node* link; // Menyimpan alamat (pointer) ke simpul berikutnya
};

int main() {
    // Deklarasi node-node
    struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
    struct Node *link, *l3ptr;

    // Inisialisasi node-node dengan menggunakan potongan kode soal
    l1.link = NULL;
    l1.alphabet = "F";

    l2.link = NULL;   // Mengatur pointer link node l2 menjadi NULL (akhir dari linked list)
    l2.alphabet = "M";  // Menyimpan huruf "M" dalam node l2

    l3.link = &l6;  // Menyambungkan node l3 ke node l6
    l3.alphabet = "A";  // Menyimpan huruf "A" dalam node l3

    l4.link = &l7;  // Menyambungkan node l4 ke node l7
    l4.alphabet = "I";  // Menyimpan huruf "I" dalam node l4

    l5.link = &l3;  // Menyambungkan node l5 ke node l3
    l5.alphabet = "K";  // Menyimpan huruf "K" dalam node l5

    l6.link = &l9;  // Menyambungkan node l6 ke node l9
    l6.alphabet = "T";  // Menyimpan huruf "T" dalam node l6

    l7.link = &l1;  // Menyambungkan node l7 ke node l1
    l7.alphabet = "N";  // Menyimpan huruf "N" dalam node l7
```

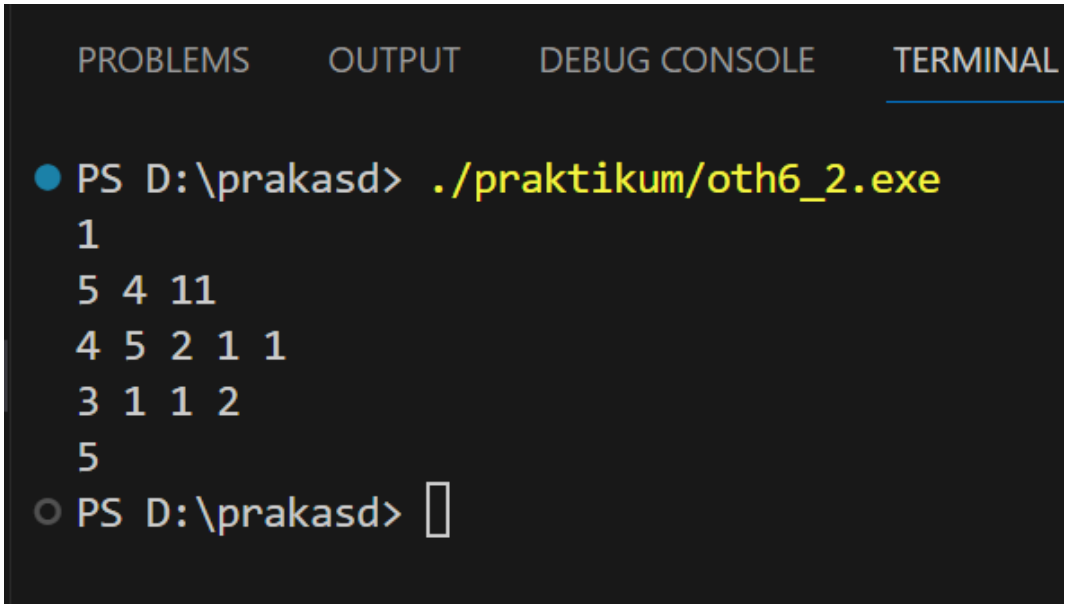
```
    l8.link = &l2;  // Menyambungkan node l8 ke node l2
    l8.alphabet = "O";  // Menyimpan huruf "O" dalam node l8

    l9.link = &l4;  // Menyambungkan node l9 ke node l4
    l9.alphabet = "R";  // Menyimpan huruf "R" dalam node l9

    // Mengatur koneksi antar node sesuai dengan urutan yang
diinginkan
    l7.link = &l1;  // Menyambungkan ke l1
    l1.link = &l8;  // Menyambungkan ke l8
    l8.link = &l2;  // Menyambungkan ke l2
    l2.link = &l5;  // Menyambungkan ke l5
    l5.link = &l3;  // Menyambungkan ke l3
    l3.link = &l6;  // Menyambungkan ke l6
    l6.link = &l9;  // Menyambungkan ke l9
    l9.link = &l4;  // Menyambungkan ke l4
    l4.link = &l7;  // Menyambungkan ke l7

    // Starting point
    l3ptr = &l7;

    // Akses data menggunakan printf
    printf("%s", l3.link->link->link->alphabet); // Menampilkan
huruf I
    printf("%s", l3.link->link->link->link->alphabet); //
Menampilkan huruf N
    printf("%s", l3.link->link->link->link->link->alphabet); //
Menampilkan huruf F
    printf("%s", l3.link->link->link->link->link->link->alphabet);
// Menampilkan huruf O
    printf("%s", l3.link->link->alphabet); // Menampilkan huruf R
    printf("%s", l3.link->link->link->link->link->link->link-
>alphabet); // Menampilkan huruf M
    printf("%s", l3.alphabet); // Menampilkan huruf A
    printf("%s", l3.link->alphabet); // Menampilkan huruf T
    printf("%s", l3.link->link->link->alphabet); // Menampilkan
huruf I
    printf("%s", l3.link->link->link->link->link->link->link->link-
>alphabet); // Menampilkan huruf K
    printf("%s", l3.alphabet); // Menampilkan huruf A

    return 0;
}
```



Gambar 1 Output soal 1

```c
#include <stdio.h>

// Fungsi untuk menghitung jumlah maksimum elemen yang dapat diambil dari kedua tumpukan
int twoStacks(int maxSum, int a[], int n, int b[], int m) {
    int sum = 0, count = 0, temp = 0, i = 0, j = 0;

    // Iterasi pertama pada tumpukan A untuk menentukan berapa banyak elemen yang dapat diambil tanpa melebihi maxSum
    while (i < n && sum + a[i] <= maxSum) {
        sum += a[i++]; // Tambahkan elemen ke sum dan naikkan indeks i
    }
    count = i; // Set count sebagai jumlah elemen yang sudah diambil dari tumpukan A

    // Iterasi kedua pada tumpukan B
    while (j < m && i >= 0) {
        sum += b[j++]; // Tambahkan elemen tumpukan B ke sum dan naikkan indeks j
        // Ketika sum melebihi maxSum, kurangi elemen dari tumpukan A sampai sum tidak melebihi maxSum
        while (sum > maxSum && i > 0) {
            sum -= a[--i];
        }
        // Jika sum masih tidak melebihi maxSum dan jumlah total elemen yang diambil dari kedua tumpukan lebih besar dari count,
        // update count dengan jumlah total elemen tersebut
        if (sum <= maxSum && i + j > count) {
```

```
            count = i + j;
        }
    }
    return count; // Kembalikan jumlah maksimum elemen yang dapat
diambil dari kedua tumpukan
}

int main() {
    int g;
    scanf("%d", &g); // Input jumlah kasus uji
    while (g--) {
        int n, m, maxSum;
        scanf("%d%d%d", &n, &m, &maxSum); // Input ukuran tumpukan
A, tumpukan B, dan maxSum
        int a[n], b[m];
        // Input elemen-elemen tumpukan A
        for (int i = 0; i < n; i++) {
            scanf("%d", &a[i]);
        }
        // Input elemen-elemen tumpukan B
        for (int i = 0; i < m; i++) {
            scanf("%d", &b[i]);
        }
        // Panggil fungsi twoStacks untuk menghitung jumlah maksimum
elemen yang dapat diambil dari kedua tumpukan
        printf("%d\n", twoStacks(maxSum, a, n, b, m));
    }
    return 0;
}
```



Gambar 2 Output soal 2