Nama : Muhammad Umar Al faruq

NIM : 1203230004

Kelas : IF 03-03

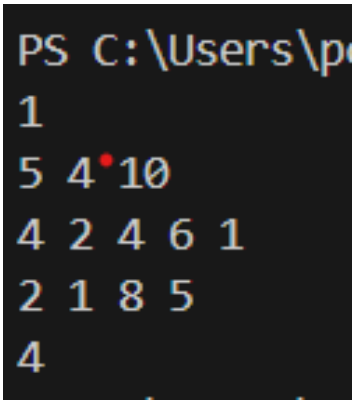
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

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

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

    // Simulate removing elements from stack a
    while (i < a_count && sum + a[i] <= maxSum) {
        sum += a[i];
        i++;
        count++;
    }

    // Simulate removing elements from stack b, while adjusting sum
    while (j < b_count && i >= 0) {
        sum += b[j];
        j++;

        // If sum exceeds maxSum, remove elements from stack a until it's
within limit
        while (sum > maxSum && i > 0) {
            i--;
            sum -= a[i];
        }

        // Update count if the current sum is within limit and greater than
previous count
        if (sum <= maxSum && i + j > count) {
            count = i + j;
        }
    }

    return count;
}
```

```c
int main() {
    int g;
    scanf("%d", &g); // Read the number of test cases

    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;
}
```

penjelasan

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

// Fungsi untuk membaca bilangan bulat ke dalam sebuah array secara dinamis
int* read_integers(int size) {
    int* arr = malloc(size * sizeof(int)); // Alokasi memori untuk array
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]); // Membaca bilangan bulat dari input dan
menyimpannya ke dalam array
    }
    return arr; // Mengembalikan pointer ke array
}

// Fungsi untuk mencari jumlah maksimum elemen yang dapat dihapus dari kedua
tumpukan
int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
    int i = 0, j = 0, count = 0, sum = 0;

    // Simulasi menghapus elemen-elemen dari tumpukan a
    while (i < a_count && sum + a[i] <= maxSum) {
        sum += a[i]; // Menambahkan elemen dari tumpukan a ke dalam jumlah
        i++; // Pindah ke elemen berikutnya di tumpukan a
        count++; // Menambah jumlah elemen yang dihapus
    }
```

```
    // Simulasi menghapus elemen-elemen dari tumpukan b, sambil menyesuaikan
jumlah
    while (j < b_count && i >= 0) {
        sum += b[j]; // Menambahkan elemen dari tumpukan b ke dalam jumlah
        j++; // Pindah ke elemen berikutnya di tumpukan b

        // Jika jumlah melebihi maxSum, hapus elemen-elemen dari tumpukan a
sampai jumlahnya dalam batas
        while (sum > maxSum && i > 0) {
            i--; // Pindah ke elemen sebelumnya di tumpukan a
            sum -= a[i]; // Mengurangi elemen yang dihapus dari tumpukan a
dari jumlah
        }

        // Memperbarui jumlah jika jumlah saat ini dalam batas dan lebih besar
dari jumlah sebelumnya
        if (sum <= maxSum && i + j > count) {
            count = i + j; // Memperbarui jumlah dengan jumlah maksimum elemen
yang dihapus sejauh ini
        }
    }

    return count; // Mengembalikan jumlah maksimum elemen yang dapat dihapus
}

int main() {
    int g;
    scanf("%d", &g); // Membaca jumlah kasus uji

    for (int g_itr = 0; g_itr < g; g_itr++) {
        int n, m, maxSum;
        scanf("%d %d %d", &n, &m, &maxSum); // Membaca ukuran tumpukan dan
maxSum untuk kasus uji saat ini

        int* a = read_integers(n); // Membaca bilangan bulat ke tumpukan a
        int* b = read_integers(m); // Membaca bilangan bulat ke tumpukan b

        int result = twoStacks(maxSum, n, a, m, b); // Mencari jumlah maksimum
elemen yang dapat dihapus
        printf("%d\n", result); // Mencetak hasil

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

    return 0; // Keluar dari program
}
```


PS C:\Users\p
1
5 4 10
4 2 4 6 1
2 1 8 5
4