Nama :rimba Berlian Aji Widiarta
NIM: 1203230111
1.

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

// Definisikan struktur Stone
struct Stone {
    char alphabet;
    struct Stone* link;
};

int main() {
    // Inisialisasi node-node sesuai dengan tabel yang diberikan
    struct Stone l1, l2, l3, l4, l5, l6, l7, l8, l9;

    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';

    // Melakukan linking antar batu sesuai urutan yang diberikan untuk membentuk kata "INFORMATIKA"
    l1.link = &l4; // A->I
    l2.link = &l7; // I->N
    l3.link = &l1; // N->F
    l4.link = &l8; // O->O
    l5.link = &l9; // F->R
    l6.link = &l2; // R->T
    l7.link = &l3; // T->A
    l8.link = &l6; // M->K
    l9.link = &l5; // K->M

    // Mencetak hasil
printf("%c%c%c%c%c%c%c%c%c%c\n", l1.link->alphabet, l2.link->alphabet, l3.link->alphabet,
        l4.link->alphabet, l5.link->alphabet, l6.link->alphabet,
        l7.link->alphabet, l8.link->alphabet, l1.link->alphabet,
        l9.link->alphabet, l7.link->alphabet);

    return 0;
}
```

```
/tmp/FQt05sqsxx.o
INFORMATIKA
```

baris 5 sampai 7

Definisikan struktur Stone dengan dua elemen: alphabet (karakter) dan link (pointer ke struktur Stone).

baris 12 sampai 39

Inisialisasi variabel l1 sampai l9 dengan tipe Stone, masing-masing dengan elemen link NULL dan alphabet dengan nilai yang berbeda.

baris 42 sampai 50

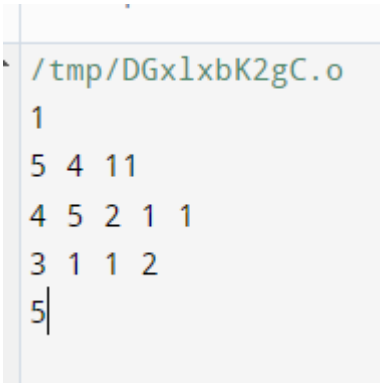
Melakukan linking antar struktur Stone menjadi sebuah linked list, sehingga membentuk kata "INFORMATIKA".

baris 53 sampai 56
Mencetak karakter dari alphabet elemen yang terhubung dalam linked list, yang akan menghasilkan kata "INFORMATIKA"

.

2.

```c
#include <stdio.h>

#define MAX(a, b) ((a) > (b) ? (a) : (b))

int twoStacks(int maxSum, int* a, int n, int* b, int m) {
    int i = 0, j = 0;
    int sum = 0, count = 0, maxCount = 0;

    // Calculate the maximum number of elements from stack a that can be selected
    while (i < n && sum + a[i] <= maxSum) {
        sum += a[i];
        i++;
        count++;
    }
    maxCount = count;

    // Now, try selecting elements from stack b and adjust the count accordingly
    while (j < m && i >= 0) {
        sum += b[j];
        j++;
        count++;

        // If sum exceeds maxSum, adjust by removing elements from stack a
        while (sum > maxSum && i > 0) {
            i--;
            sum -= a[i];
            count--;
        }

        // Update maxCount if necessary
        if (sum <= maxSum)
            maxCount = MAX(maxCount, count);
    }

    return maxCount;
}

int main() {
    int g;
    scanf("%d", &g);

    while (g--) {
        int n, m, x;
        scanf("%d %d %d", &n, &m, &x);

        int a[n], b[m];
        for (int i = 0; i < n; i++)
            scanf("%d", &a[i]);
        for (int i = 0; i < m; i++)
            scanf("%d", &b[i]);

        printf("%d\n", twoStacks(x, a, n, b, m));
    }

    return 0;
}
```



baris 5
Ini adalah deklarasi fungsi twoStacks(). Fungsi ini akan menerima lima parameter: maxSum (nilai maksimum yang dapat diambil), a (array yang merepresentasikan tumpukan pertama), n (ukuran tumpukan pertama), b (array yang merepresentasikan tumpukan kedua), dan m (ukuran tumpukan kedua).

baris 6 sampai 7
Deklarasi beberapa variabel lokal yang akan digunakan di dalam fungsi

baris 10 sampai 15
Loop ini menghitung jumlah maksimum elemen yang dapat diambil dari tumpukan pertama (a) sedemikian rupa sehingga jumlahnya tidak melebihi maxSum.

baris 18 sampai 21
Loop ini mencoba memilih elemen dari tumpukan kedua (b) dan menyesuaikan jumlahnya. Ini dilakukan selama masih ada elemen yang tersedia di tumpukan kedua (j < m) dan masih ada elemen yang tersisa di tumpukan pertama (i >= 0)

baris 24 sampai 27
Pada bagian ini, dilakukan pengecekan apakah sum (jumlah elemen yang sudah dipilih dari kedua tumpukan) melebihi maxSum. Jika melebihi, maka dilakukan penyesuaian dengan menghapus beberapa elemen dari tumpukan pertama (a) yang telah dipilih sebelumnya

baris 31 sampai 32
Pada bagian ini, dilakukan pengecekan apakah sum (jumlah elemen yang sudah dipilih dari kedua tumpukan) melebihi maxSum. Jika melebihi, maka dilakukan penyesuaian dengan menghapus beberapa elemen dari tumpukan pertama (a) yang telah dipilih sebelumnya

baris 35
Fungsi twoStacks() mengembalikan jumlah maksimum elemen yang dapat dipilih dari kedua tumpukan.

baris 38 sampai 40
Fungsi main() dimulai di sini. Program ini membaca sebuah integer yang menunjukkan jumlah kasus uji.

baris 42
Ini adalah loop while yang akan berjalan selama nilai dari variabel g positif. g-- digunakan untuk mengurangi nilai g setiap kali loop dieksekusi, sehingga loop akan berhenti setelah g menjadi nol atau negatif

baris 43 sampai 44
Baris ini membaca tiga integer dari input, yaitu n, m, dan x, yang masing-masing mewakili ukuran tumpukan pertama (a), ukuran tumpukan kedua (b), dan nilai maksimum yang dapat diambil dari kedua tumpukan.

baris 46
Baris ini mendeklarasikan dua array, a dan b, dengan ukuran sesuai dengan nilai yang dibaca sebelumnya (n dan m).
.
baris 47 sampai 48
Loop ini membaca n integer yang mewakili elemen-elemen dari tumpukan pertama (a) dan menyimpannya ke dalam array a.

baris 49 sampai 50
Loop ini mirip dengan yang sebelumnya, tetapi kali ini membaca m integer untuk tumpukan kedua (b) dan menyimpannya ke dalam array b.

baris 52
Baris ini memanggil fungsi twoStacks dengan argumen x (nilai maksimum yang dapat diambil), array a (tumpukan pertama), ukuran n dari tumpukan pertama, array b (tumpukan kedua), dan ukuran m dari tumpukan kedua.