NAMA : CLAURA EVELYNE CHENTYA ANNELY SIAGIAN

KELAS : IF 03 – 02

NIM : 1203230078

NO 1

SOURCE CODE

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

//Strucktur node
struct Node {
    char* alphabet;
    struct Node* link;
};

int main() {
    // Deklarasi node
    struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9; //Variabel node berjumalah
9
    struct Node *link, *l3ptr;

    // Inisialisasi node node yang menggunakan potongan dari kode soal
    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";

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

    // Starting point
    l3ptr = &l7;

    // Mengakses 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->link-
>alphabet);// Menampilkan huruf K
    printf("%s", l3.alphabet);// Menampilkan huruf A

    return 0;
}
```

OUTPUT

```
PS C:\Users\ASUS> cd "C:\Users\ASUS\AppData
INFORMATIKA
PS C:\Users\ASUS\AppData\Local\Temp>
```

PENJELASAN

```
//Strucktur node
struct Node {
    char* alphabet;
    struct Node* link;
};
```

struktur Node memiliki dua anggota, yaitu alphabet yang merupakan pointer ke karakter dan link yang merupakan pointer ke struktur Node lainnya. Struktur ini akan digunakan untuk membuat linked list.

```
int main() {
    // Deklarasi Node
    struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9; //Variabel Node berjumlah
9
    struct Node *link, *l3ptr;
```

Beberapa variabel yang akan digunakan dalam program. Variabel l1 - l9 adalah variabel bertipe struct Node yang akan digunakan untuk menyimpan data dalam linked list. Variabel link dan l3ptr adalah pointer ke struct Node.

```
 // Inisialisasi node node yang menggunakan potongan dari kode soal
    l1.link = NULL;
    l1.alphabet = "F";

    l2.link = NULL;
    l2.alphabet = "M";

    l3.link = NULL;
    l3.alphabet = "A";
```

Dst. Menginisialisasi nilai-nilai dari variabel l1 - l9. Setiap variabel l1 hingga l9 memiliki anggota link yang diatur menjadi NULL dan anggota alphabet yang diatur menjadi karakter tertentu.

```
 // Mengatur koneksi antar node yang lain sesuai dengan urutan yang diinginkan
dan menyambungkannya ke l1
    l7.link = &l1;
    l1.link = &l8;
    l8.link = &l2;
    l2.link = &l5;
```

Mengatur koneksi antara variabel l1 hingga l9 sesuai dengan urutan yang diinginkan.

```
 // Starting point
    l3ptr = &l7;
```

Mengatur variabel l3ptr menjadi pointer ke variabel l7. Variabel l3ptr digunakan sebagai titik awal untuk mengakses data dalam linked list.

```
  // Mengakses 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
```

Fungsi printf untuk mengakses dan menampilkan data dalam linked list. Setiap printf mengakses anggota alphabet dari variabel l3 dan variabel link yang terhubung dengan l3 menggunakan operator ->. Hasilnya adalah menampilkan karakter-karakter tertentu sesuai dengan urutan yang diakses.

NO 2

SOURCE CODE

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

#define MAX_SIZE 100005

int twoStacks(int maxSum, int a[], int n, int b[], int m) {
    int stackA[MAX_SIZE], stackB[MAX_SIZE];
    int topA = -1, topB = -1;
    int sum = 0, count = 0;
    int i = 0, j = 0;

    while (i < n && sum + a[i] <= maxSum) {
        sum += a[i];
        stackA[++topA] = a[i++];
    }

    count = topA + 1;

    while (j < m && i >= 0) {
        sum += b[j];
        stackB[++topB] = b[j++];

        while (sum > maxSum && topA >= 0) {
            sum -= stackA[topA--];
        }

        if (sum <= maxSum && topA + topB + 2 > count) {
            count = topA + topB + 2;
```

```
        }
    }

    return count;
}

int main() {
    int g;
    scanf("%d", &g);
    while (g--) {
        int n, m, maxSum;
        scanf("%d%d%d", &n, &m, &maxSum);
        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(maxSum, a, n, b, m));
    }
    return 0;
}
```

OUTPUT

```
PS C:\Users\ASUS> cd "C:\Users\ASUS\AppDat
1
5 4 11
4 5 2 1 1
3 1 1 2
5
PS C:\Users\ASUS\AppData\Local\Temp>
```

Penjelasan

define MAX_SIZE 100005: Ukuran maksimum untuk array yang digunakan dalam program.

int twoStacks(int maxSum, int a[], int n, int b[], int m): Fungsi yang mengambil maksimum jumlah yang diizinkan (`maxSum`), dua array integer (`a[]` dan `b[]`), serta panjang masing-masing array (`n` dan `m`). Fungsi ini akan mengembalikan jumlah maksimum elemen yang dapat diambil dari kedua tumpukan sehingga total nilai elemen yang diambil tidak melebihi `maxSum`.

int stackA[MAX_SIZE], stackB[MAX_SIZE]; : Mendeklarasikan dua array sebagai tumpukan (stackA dan stackB) dengan ukuran maksimum `MAX_SIZE`.

int topA = -1, topB = -1; : Dua variabel integer sebagai indeks puncak (top) dari masing-masing tumpukan, dan diatur ke -1.

int sum = 0, count = 0; : Untuk menyimpan total nilai elemen yang sudah diambil, dan `count` untuk menyimpan jumlah elemen yang telah diambil.

int i = 0, j = 0; : Dua variabel integer sebagai indeks untuk array `a[]` dan `b[]`, awalnya diatur ke 0.

while (i < n && sum + a[i] <= maxSum) { ... }: Looping untuk mengambil elemen dari tumpukan A (`a[]`) hingga nilai elemen yang sudah diambil tidak melebihi `maxSum`.

count = topA + 1; : Jumlah elemen yang telah diambil dari tumpukan A dihitung.

while (j < m && i >= 0) { ... }: Untuk menambahkan elemen dari tumpukan B (`b[]`) ke dalam tumpukan dan memperbarui total nilai elemen yang sudah diambil.

while (sum > maxSum && topA >= 0) { ... } : Untuk menghapus elemen dari tumpukan A jika total nilai elemen yang sudah diambil melebihi `maxSum`.

if (sum <= maxSum && topA + topB + 2 > count) { ... } : Memperbarui `count` kalau jumlah total elemen yang diambil dari kedua tumpukan lebih besar dari nilai sebelumnya.

return count; : Mengembalikan nilai `count` yang jumlah maksimum elemen yang dapat diambil dari kedua tumpukan.

while (g--) { ... }: Looping untuk setiap kasus uji.

scanf("%d%d%d", &n, &m, &maxSum); : Meminta input panjang tumpukan A (`n`), panjang tumpukan B (`m`), dan nilai maksimum yang diizinkan (`maxSum`).

for (int i = 0; i < n; i++) { ... }: Looping untuk membaca elemen-elemen tumpukan A

for (int i = 0; i < m; i++) { ... }: Looping untuk membaca elemen-elemen tumpukan B

printf("%d\n", twoStacks(maxSum, a, n, b, m)); : Mencetak hasil pemrosesan dari fungsi `twoStacks()`.