

Nama : muamar haikal fauzan

Nim : 1203230118

IF 03-02

SOURCE CODE

1.

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

typedef struct Node {
    char *alphabet;
    struct Node *link;
} Node;

int main() {

    Node l1 = {.link = NULL, .alphabet = "F"};
    Node l2 = {.link = NULL, .alphabet = "M"};
    Node l3 = {.link = NULL, .alphabet = "A"};
    Node l4 = {.link = NULL, .alphabet = "I"};
    Node l5 = {.link = NULL, .alphabet = "K"};
    Node l6 = {.link = NULL, .alphabet = "T"};
    Node l7 = {.link = NULL, .alphabet = "N"};
    Node l8 = {.link = NULL, .alphabet = "O"};
    Node l9 = {.link = NULL, .alphabet = "R"};

    l3.link = &l4;
    l4.link = &l5;
    l5.link = &l6;
    l6.link = &l7;
    l7.link = &l8;
    l8.link = &l9;
    l9.link = &l1;
    l1.link = &l2;

    printf("%s", l3.link->alphabet);
```

```

printf("%s", l3.link->link->link->link->alphabet);
printf("%s", l3.link->link->link->link->link->link->link->alphabet);
printf("%s", l3.link->link->link->link->link->alphabet);
printf("%s", l3.link->link->link->link->link->link->alphabet);
printf("%s", l3.link->link->link->link->link->link->link->link->alphabet);
printf("%s", l3.alphabet);
printf("%s", l3.link->link->link->alphabet);
printf("%s", l3.link->alphabet);
printf("%s", l3.link->link->alphabet);
printf("%s", l3.alphabet);

return 0;
}

```

2. HACKERRANK

```

#include <stdio.h>

int twoStacks(int maxSum, int n, int m, int *a, int *b) {
    int i = 0, j = 0, count = 0, sum = 0;
    while (i < n && sum + a[i] <= maxSum) {
        sum += a[i++];
        count++;
    }
    int maxCount = count;
    while (j < m && i >= 0) {
        sum += b[j++];
        count++;
        while (sum > maxSum && i > 0) {
            sum -= a[--i];
            count--;
        }
        if (sum <= maxSum && count > maxCount)
            maxCount = count;
    }
}

```

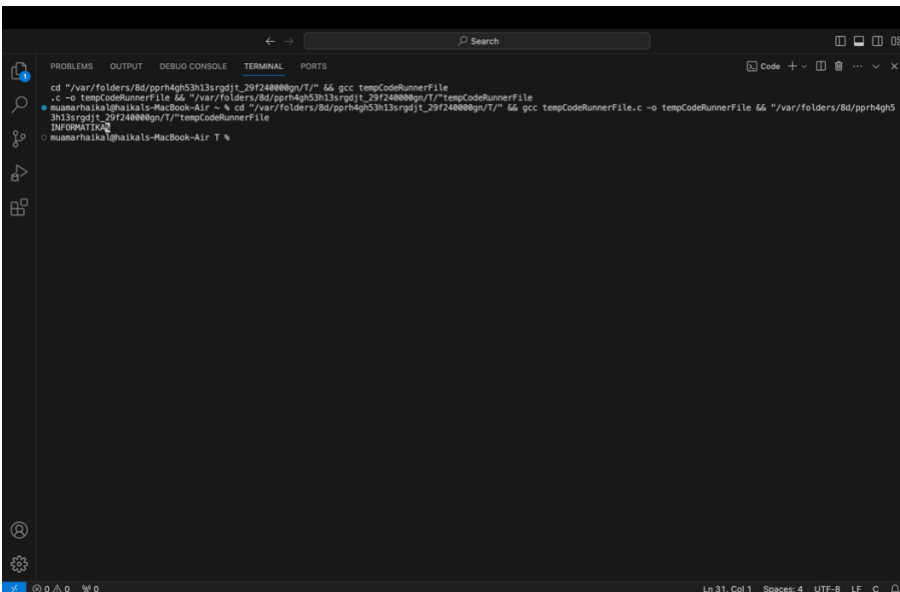
```

    }
    return maxCount;
}

int main() {
    int games;
    scanf("%d", &games);
    while (games--) {
        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]);
        int result = twoStacks(maxSum, n, m, a, b);
        printf("%d\n", result);
    }
    return 0;
}

```

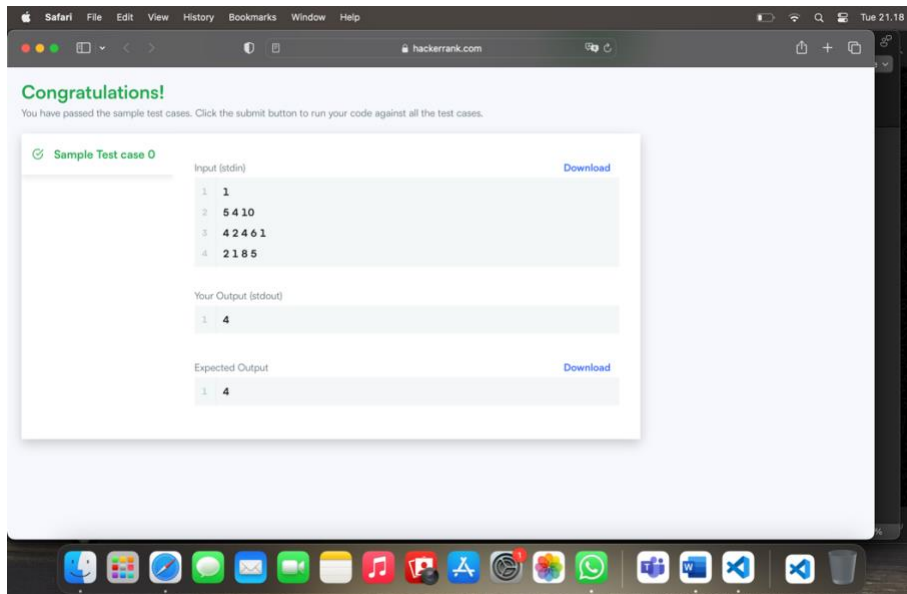
HASIL OUTPUT



```

1. cd "/var/folders/8d/pprh4gh53h13srgdjt_29f240000gn/T/" && gcc tempCodeRunnerFile
   c -o tempCodeRunnerFile && "/var/folders/8d/pprh4gh53h13srgdjt_29f240000gn/T/"tempCodeRunnerFile
   • muamrhaika@haikals-MacBook-Air ~ % cd "/var/folders/8d/pprh4gh53h13srgdjt_29f240000gn/T/" && gcc tempCodeRunnerFile.c -o tempCodeRunnerFile && "/var/folders/8d/pprh4gh5
   3h13srgdjt_29f240000gn/T/"tempCodeRunnerFile
   Th999AT112
   ○ muamrhaika@haikals-MacBook-Air T %

```



2.

Penjelasan code

Soal 1.

- Struktur Node didefinisikan dengan dua anggota: `alphabet` (pointer ke karakter) dan `link` (pointer ke node berikutnya dalam linked list).
- Node-node `l1` sampai `l9` dideklarasikan dan diinisialisasi dengan `link` yang menunjuk ke `NULL` dan `alphabet` berisi karakter yang diberikan.
- Node-node tersebut dihubungkan satu sama lain dengan mengatur `link` masing-masing node sehingga membentuk linked list sirkular, dimana node terakhir mengarah kembali ke node pertama.
- Kemudian, isi beberapa node dalam linked list dicetak menggunakan `printf`. Misalnya, `l3.link->alphabet` mencetak karakter dari node yang dihubungkan dengan `l3`, dan seterusnya.

Soal 2.

- **Fungsi twoStacks:**
 - Fungsi ini menerima argumen `maxSum` (nilai maksimum yang diperbolehkan), `n` (jumlah elemen dalam tumpukan `a`), `m` (jumlah elemen dalam tumpukan `b`), dan pointer ke array `a` dan `b`.
 - Variabel `i`, `j`, dan `count` digunakan untuk melacak posisi saat ini dalam array dan jumlah elemen yang diambil dari tumpukan.
 - Selama elemen dari tumpukan `a` masih dapat ditambahkan ke `sum` tanpa melebihi `maxSum`, elemen tersebut diambil dan `count` diinkrementasi.

- Variabel `maxCount` menyimpan jumlah maksimum elemen yang berhasil diambil.
- Selanjutnya, elemen dari tumpukan `b` ditambahkan ke `sum` satu per satu. Jika `sum` melebihi `maxSum`, elemen-elemen dari tumpukan `a` dikurangi dari `sum` sampai `sum` tidak melebihi `maxSum`.
- Pada setiap iterasi, jika jumlah elemen yang diambil (`count`) lebih besar dari `maxCount`, `maxCount` diperbarui.
- Fungsi mengembalikan `maxCount`.
- **Fungsi main:**
 - Di dalam fungsi `main`, pertama-tama dibaca jumlah permainan yang akan diuji menggunakan `scanf`.
 - Kemudian, dilakukan iterasi sebanyak jumlah permainan tersebut.
 - Dalam setiap iterasi, dibaca jumlah elemen dalam tumpukan `a` (`n`), tumpukan `b` (`m`), dan nilai maksimum `maxSum`.
 - Selanjutnya, elemen-elemen dari tumpukan `a` dan `b` dibaca menggunakan `scanf`.
 - Panggilan fungsi `twoStacks` dilakukan dengan argumen yang sesuai, dan hasilnya dicetak.
 - Proses ini diulangi hingga semua permainan diuji.
- **Input dan Output:**
 - Program ini mengharapkan input dari pengguna yang berisi jumlah permainan, diikuti oleh jumlah elemen dalam tumpukan `a` dan `b`, nilai maksimum `maxSum`, dan elemen-elemen dari kedua tumpukan.
 - Untuk setiap permainan, program mencetak jumlah maksimum elemen yang dapat diambil dari kedua tumpukan tanpa melebihi `maxSum`.