Nama: I Gede Hermawan Adi Pranata
Nim: 1203230029
Kelas: IF 03-01

1. Tugasmu kali ini adalah memecahkan teka-teki berantai yang tersusun dalam serangkaian batu yang tersebar di dalam hutan. Petunjuk-petunjuk untuk menyelesaikan teka-teki ini tersimpan dalam urutan batu-batu yang saling terhubung dengan tanda panah, membentuk pola yang harus diikuti untuk menemukan solusi. Berikut merupakan rangkaian batu yang telah kamu amati.
   Pada setiap batu tersebut saling terhubung dengan tanda panah yang dimulai dari batu dengan huruf “N” dan seterusnya mengikuti arah panah. Ternyata, jika kita mengikuti arah panah dari batu dengan huruf "N", dan menggabungkan huruf-huruf pada batu-batu tersebut, kita akan membentuk sebuah kata yang sangat penting: "INFORMATIKA". Tapi tunggu, Sherlock Holmes memberimu sebuah catatan dalam bentuk kode yang akan membantumu menyelesaikan teka-teki ini. Kode tersebut menggunakan konsep Self Referential Structures untuk merepresentasikan hubungan antar batu-batu.
   Tugasmu sekarang adalah merangkai huruf tersebut hingga membentuk suatu kata yang telah ditentukan dengan menggunakan Self Referential Structures dan ketentuan sebagai berikut.
   1. Wajib menggunakan potongan kode yang ada pada tabel diatas untuk inisialisasi (tidak ada penambahan / pengurangan ketika inisialisasi).
   2. Pastikan ketika membuat linking / koneksi harus sesuai dengan gambar diatas (urutan dimulai dari huruf N dan mengikuti arah panah).
   3. Lakukan semua akses data dengan menggunakan l3 sebagai starting point.

- **Code**

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

struct Stone {
    char* alphabet ;
    struct Stone *link;
};

int main(){
    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";

l1.link=&l8; //F ke O
l8.link=&l2; //O ke M
l2.link=&l5; //M ke K
l5.link=&l3; //K ke A
l3.link=&l6; //A ke T
l6.link=&l9; //T ke r
l9.link=&l4; //R ke I
l4.link=&l7; //I ke N
l7.link=&l1; //N ke F
//akses data
 printf("%s",l3.link->link->link->alphabet);
 printf("%s",l3.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->link->link->link->
>alphabet);
 printf("%s",l3.link->link->link->link->link->link->link->link->link->link->link->
>link->link->link->link->link->alphabet);
 printf("%s",l3.link->link->link->link->link->link->link->link->link->link->link->
>link->link->link->link->link->link->link->alphabet);
 printf("%s",l3.link->link->link->link->link->link->link->link->link->link->link->
>link->link->link->link->link->link->link->link->alphabet);
 printf("%s",l3.link->link->link->link->link->link->link->link->link->link->link->
>link->link->link->link->link->link->link->link->link->alphabet);
  printf("%s",l3.link->link->link->link->link->link->link->link->link->link->link->
>link->link->link->link->link->link->link->link->link->link->link->link->link->
>link->link->alphabet);
  printf("%s",l3.link->link->link->link->link->link->link->link->link->link->link->
>link->link->link->link->link->link->link->link->link->link->link->link->link->
>link->link->link->link->alphabet);
struct Stone* currnt=&l3;
return 0;
}
```

- **Penjelasan**

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

Baris ini adalah direktif preprosesor yang mengarahkan compiler untuk memasukkan kode dari file header standar <stdio.h> dan <stdlib.h>, yang berisi deklarasi fungsi-fungsi standar yang digunakan dalam program ini.

```
struct Stone {
    char* alphabet ;
    struct Stone *link;
};
```

Di sini, sebuah struktur Stone dideklarasikan. Struktur ini memiliki dua anggota: sebuah pointer ke char yang disebut alphabet, dan pointer ke struct Stone yang disebut link. Ini adalah dasar untuk membuat daftar terkait.

```
int main(){
    struct Stone l1,l2,l3,l4,l5,l6,l7,l8,l9;
```

Deklarasi variabel l1 hingga l9 yang bertipe struct Stone. Ini adalah simpul-simpul dalam daftar terkait.

```
l1.link = NULL;
l1.alphabet = "F";
l2.link = NULL;
l2.alphabet = "M";
l3.link = NULL;
l3.alphabet = "A";
l4.link = NULL;
 . . . .
```

Setiap variabel l1 hingga l9 diinisialisasi. Setiap simpul memiliki alphabet yang menunjukkan sebuah huruf dan link yang menunjukkan simpul berikutnya dalam daftar terkait. Nilai NULL diberikan ke link untuk semua simpul saat ini.

```
l1.link=&l8; //F ke O
l8.link=&l2; //O ke M
l2.link=&l5; //M ke K
 . . . .
```

Selanjutnya, diatur hubungan antara simpul-simpul dalam daftar terkait. Contohnya, l1 dihubungkan ke l8, yang berarti simpul l1 menunjuk ke simpul l8.

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

Bagian ini mencetak data dari simpul-simpul dalam daftar terkait. Misalnya, l3.link->link->link->alphabet berarti mengakses huruf dari simpul keempat dalam daftar (dari simpul l3, melalui simpul-simpul yang terhubung setelahnya).

```
struct Stone* currnt=&l3;
```

Sebuah pointer currnt ditetapkan ke alamat dari simpul l3. Ini dapat digunakan untuk mengakses simpul ini atau simpul lain dalam daftar terkait.

```
return 0;
```

Ini adalah keluaran dari fungsi main(), menunjukkan bahwa program selesai dijalankan

- **Output**

```
INFORMATIK
PS D:\code\c tugas> cd "d:\code\c tugas\" ; if ($?)
ugasasd11 } ; if ($?) { .\tugasasd11 }
INFORMATIKA
PS D:\code\c tugas>
```

2. Selesaikan soal pada link di bawah ini!
https://www.hackerrank.com/challenges/game-of-two-stacks/problem

- **Code**

```c
#include <stdio.h>

#define MAX_N 100000
#define MAX_M 100000

int twoStacks(int maxSum, int a_count, int a[], int b_count, int b[]) {
    int sum = 0, count = 0, index_a = 0, index_b = 0;

    while (index_a < a_count && sum + a[index_a] <= maxSum) {
        sum += a[index_a++];
        count++;
    }

    int maxCount = count;

    while (index_b < b_count && index_a > 0) {
        sum += b[index_b++];

        while (sum > maxSum && index_a > 0) {
```

```
            sum -= a[--index_a];
            count--;
        }

        if (sum <= maxSum && count > maxCount)
            maxCount = count;
    }

    return maxCount;
}

int main() {
    int g;
    printf("Enter the number of games: ");
    scanf("%d", &g);

    while (g--) {
        int n, m, maxSum;
        printf("Enter n, m, and maxSum: ");
        scanf("%d %d %d", &n, &m, &maxSum);

        int a[MAX_N], b[MAX_M];

        printf("Enter elements of stack a: ");
        for (int i = 0; i < n; i++)
            scanf("%d", &a[i]);

        printf("Enter elements of stack b: ");
        for (int i = 0; i < m; i++)
            scanf("%d", &b[i]);

        printf("Maximum possible score: %d\n", twoStacks(maxSum, n, a, m, b));
    }

    return 0;
}
```

- **Penjelasan**

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

Baris ini memasukkan library standar stdio.h yang berisi fungsi-fungsi untuk input-output standar.

```
#define MAX_N 100000
#define MAX_M 100000
```

Baris ini mendefinisikan dua konstanta, MAX_N dan MAX_M, yang masing-masing merepresentasikan jumlah maksimum elemen dalam tumpukan a dan b.

```
int twoStacks(int maxSum, int a_count, int a[], int b_count, int b[]) {
```

Ini adalah definisi fungsi twoStacks yang mengambil lima argumen: maxSum (batas jumlah maksimum), a_count (jumlah elemen dalam tumpukan a), a (array yang berisi elemen-elemen tumpukan a), b_count (jumlah elemen dalam tumpukan b), dan b (array yang berisi elemen-elemen tumpukan b). Fungsi ini mengembalikan jumlah maksimum elemen yang dapat diambil dari kedua tumpukan.

```
int sum = 0, count = 0, index_a = 0, index_b = 0;
```

Deklarasi dan inisialisasi variabel lokal sum (total jumlah elemen yang telah diambil), count (jumlah elemen yang telah diambil), index_a (indeks saat ini di tumpukan a), dan index_b (indeks saat ini di tumpukan b).

```
    while (index_a < a_count && sum + a[index_a] <= maxSum) {
        sum += a[index_a++];
        count++;
    }
```

Loop ini menambahkan elemen-elemen dari tumpukan a ke dalam sum sampai jumlahnya melebihi maxSum atau tumpukan a telah habis. Setiap elemen yang ditambahkan, variabel count akan bertambah satu.

```
    int maxCount = count;
```

Variabel maxCount digunakan untuk menyimpan jumlah maksimum elemen yang telah diambil dari kedua tumpukan.

```
while (index_b < b_count && index_a > 0) {
```

Loop ini berjalan selama masih ada elemen dalam tumpukan b dan masih ada elemen dalam tumpukan a yang bisa diambil.

```
        sum += b[index_b++];
```

Elemen-elemen dari tumpukan b ditambahkan ke dalam sum.

```
        while (sum > maxSum && index_a > 0) {
            sum -= a[--index_a];
            count--;
        }
```

Loop ini berjalan selama sum melebihi maxSum, dan elemen-elemen dari tumpukan a yang telah diambil sebelumnya akan dikeluarkan satu per satu sampai sum tidak melebihi maxSum.

```
        if (sum <= maxSum && count > maxCount)
            maxCount = count;
    }
```

Jika sum tidak melebihi maxSum dan jumlah elemen yang telah diambil lebih besar dari maxCount, maka maxCount diperbarui dengan nilai count.

```
    return maxCount;
```

Fungsi mengembalikan maxCount, yaitu jumlah maksimum elemen yang dapat diambil dari kedua tumpukan sehingga total jumlahnya tidak melebihi maxSum.

```
int main() {
    int g;
    printf("Enter the number of games: ");
    scanf("%d", &g);
```

Variabel g digunakan untuk menyimpan jumlah permainan yang akan dimainkan. Pengguna diminta untuk memasukkan jumlah permainan.

```
    while (g--) {
```

Loop ini berjalan sebanyak g kali, yang mana g adalah jumlah permainan yang akan dimainkan.

```
int n, m, maxSum;
        printf("Enter n, m, and maxSum: ");
        scanf("%d %d %d", &n, &m, &maxSum);
```

Variabel n, m, dan maxSum digunakan untuk menyimpan jumlah elemen dalam tumpukan a, jumlah elemen dalam tumpukan b, dan maxSum untuk setiap permainan. Pengguna diminta untuk memasukkan nilai-nilai ini.

```
        int a[MAX_N], b[MAX_M];
```

Deklarasi array a dan b dengan ukuran maksimum sesuai dengan konstanta yang telah didefinisikan sebelumnya.

```
printf("Enter elements of stack a: ");
        for (int i = 0; i < n; i++)
            scanf("%d", &a[i]);

        printf("Enter elements of stack b: ");
        for (int i = 0; i < m; i++)
            scanf("%d", &b[i]);
```

Pengguna diminta untuk memasukkan elemen-elemen tumpukan a dan b.

```
        printf("Maximum possible score: %d\n", twoStacks(maxSum, n, a, m, b));
    }
```

Mencetak jumlah maksimum elemen yang dapat diambil dari kedua tumpukan sehingga total jumlahnya tidak melebihi maxSum.

```
    return 0;
```

Fungsi main mengembalikan nilai 0 yang menandakan bahwa program telah berakhir

- Output

```
($?) { .\tugasasd12 }
Enter the number of games: 1
Enter n, m, and maxSum: 5 4 11
Enter elements of stack a: 4 5 2 1 1
Enter elements of stack b: 3 1 1 2
Maximum possible score: 3
PS D:\code\c tugas>
```