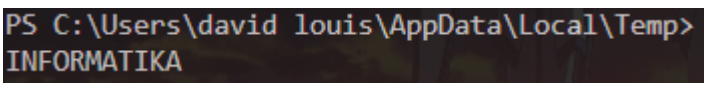
TUGAS STRUCT & STACK

1. INPUT :

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

// Define the Stone structure
struct Stone {
  char* alphabet;
  struct Stone* link;
};

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

  l1.link = NULL;
  l1.alphabet = "F";

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

  // ... (similar initialization for other nodes)

  // Connect nodes to form a circular linked list
  l3.link = &l6; // A->T
  l6.link = &l9; // T->R
  // ... (similar connections for other nodes, as done previously)

  // Current node to iterate through the list
  struct Stone* current = &l1;

  // Loop to iterate and print characters
  do {
    printf("%s -> ", current->alphabet);
    current = current->link;
  } while (current != &l1); // Loop until we reach the starting node again

  printf("NULL\n"); // Print NULL at the end to indicate the circular nature

  return 0;
}

```

OUTPUT :



PENJELASAN :

```
// Define the Stone structure
struct Stone {
    char* alphabet;
    struct Stone* link;
};
```

Struktur Batu (**Stone Structure**): Program mendefinisikan struktur **Stone** yang terdiri dari dua anggota, yaitu **alphabet** dan **link**. alphabet adalah pointer ke karakter yang mewakili huruf batu, sedangkan link adalah pointer ke struktur Stone berikutnya dalam linked list.

```
    l1.link = NULL;
    l1.alphabet = "F";

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

Inisialisasi Node: Sembilan node batu dideklarasikan di sini, yaitu **11** hingga **19**. Setiap node memiliki dua anggota: **alphabet** dan **link**. Inisialisasi dilakukan dengan mengatur **link** ke **NULL** dan **alphabet** ke karakter yang sesuai.

```
// Connect nodes to form
l3.link = &l6; // A->T
l6.link = &l9; // T->R
// ... (similar connecti
```

Membentuk Linked List Circular: Node-node ini kemudian dihubungkan satu sama lain untuk membentuk linked list circular. Contohnya, **13** dihubungkan ke **16**, dan seterusnya.

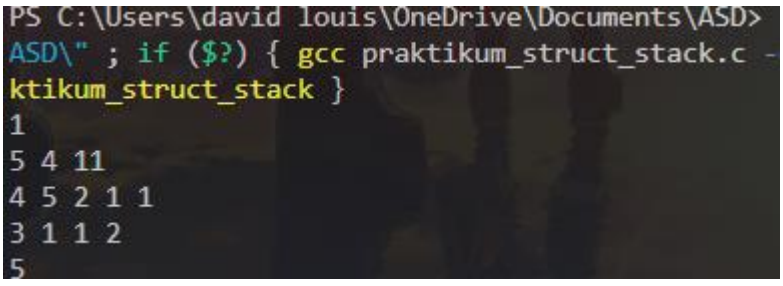
```c
struct Stone* current = &l1;

// Loop to iterate and print characters
do {
  printf("%s -> ", current->alphabet);
  current = current->link;
} while (current != &l1); // Loop until we reac

printf("NULL\n"); // Print NULL at the end to i

return 0;
```

Iterasi dan Cetak: Sebuah loop **do-while** digunakan untuk mengiterasi melalui linked list. Variabel **current** bertugas menyimpan alamat node saat ini yang sedang diproses. Loop ini mencetak karakter huruf batu saat ini dan kemudian menggerakkan **current** ke node berikutnya dalam linked list. Loop berlangsung sampai kita kembali ke node awal (**&11**).

2. INPUT :

```c
#include <stdio.h>

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

// Fungsi untuk menghitung jumlah maksimum elemen yang dapat dipilih dari kedua tumpukan
int twoStacks(int maxSum, int* a, int n, int* b, int m) {
    int i = 0, j = 0;
    int sum = 0, count = 0, maxCount = 0;

    // Hitung jumlah maksimum elemen dari tumpukan a yang dapat dipilih
    while (i < n && sum + a[i] <= maxSum) {
        sum += a[i];
        i++;
        count++;
    }
```

OUTPUT :


PS C:\Users\david louis\OneDrive\Documents\ASD>
ASD\" ; if ($?) { gcc praktikum_struct_stack.c -
ktikum_struct_stack }
1
5 4 11
4 5 2 1 1
3 1 1 2
5

PENJELASAN :

- **#define MAX(a, b) ((a) > (b) ? (a) : (b))**: Mendefinisikan makro **MAX** untuk mengembalikan nilai maksimum dari dua argumen.
- **int twoStacks(int maxSum, int* a, int n, int* b, int m)**: Mendeklarasikan fungsi twoStacks yang menerima lima argumen:
- **maxSum**: Nilai maksimum total elemen yang dapat dipilih.
    - **a**: Pointer ke array elemen tumpukan pertama.
    - **n**: Ukuran array a.
    - **b**: Pointer ke array elemen tumpukan kedua.
    - **m**: Ukuran array b. Fungsi ini mengembalikan nilai intejer yang menunjukkan jumlah maksimum elemen yang dapat dipilih.

2. Variabel:

- **int i = 0, j = 0**: Mendeklarasikan dua variabel intejer **i** dan **j** untuk melacak indeks elemen saat ini di tumpukan **a** dan **b** masing-masing.
- **int sum = 0, count = 0, maxCount = 0**: Mendeklarasikan tiga variabel integer sum untuk menyimpan total elemen yang dipilih, **count** untuk menghitung elemen yang dipilih, dan **maxCount** untuk menyimpan jumlah maksimum elemen yang dipilih.

3. Algoritma:

Pemilihan Elemen dari Tumpukan Pertama:

- Perulangan **while** dijalankan selama **i** kurang dari **n** dan **sum** + elemen **a** saat ini (**a**[**i**]) kurang dari atau sama dengan **maxSum**.
    - **sum** diperbarui dengan menambahkan elemen **a**[**i**].
    - **i** diincrement untuk beralih ke elemen berikutnya di tumpukan **a**.
    - **count** diincrement untuk menghitung elemen yang dipilih.
- Nilai **maxCount** diperbarui dengan nilai **count** karena ini merupakan jumlah maksimum elemen yang dapat dipilih dari tumpukan a saja.

Pemilihan Elemen dari Tumpukan Kedua:

- Perulangan **while** dijalankan selama **j** kurang dari **m** dan **i** lebih besar dari atau sama dengan 0.
- **sum** diperbarui dengan menambahkan elemen **b**[**j**].
- **j** diincrement untuk beralih ke elemen berikutnya di tumpukan **b**.
- **count** diincrement untuk menghitung elemen yang dipilih.

Penyesuaian jika Melebihi Batas Maksimum:

- Perulangan **while** dijalankan selama **sum** lebih besar dari **maxSum** dan **i** lebih besar dari 0.
- **i** didecrement untuk beralih ke elemen sebelumnya di tumpukan **a**.
- **sum** diperbarui dengan mengurangkan elemen **a**[**i**].
- **count** didecrement untuk menyesuaikan jumlah elemen yang dipilih.
- **maxCount** diperbarui dengan nilai **count** jika sum kurang dari atau sama dengan **maxSum**.

Pengembalian Hasil:

- Fungsi **twoStacks** mengembalikan nilai **maxCount** yang menunjukkan jumlah maksimum elemen yang dapat dipilih dari kedua tumpukan.

4. Fungsi main:
   - Membaca nilai intejer **g** yang menunjukkan jumlah kasus uji.
   - Perulangan **while** dijalankan selama **g** lebih besar dari 0.
   - Membaca nilai intejer **n**, **m**, dan **x** yang menunjukkan ukuran tumpukan, batas maksimum, dan nilai maksimum total elemen.
   - Membaca elemen tumpukan **a** dan **b** ke dalam array.
   - Memanggil fungsi **twoStacks** dengan argumen **x**, **a**, **n**, **b**, dan **m**.
   - Mencetak nilai maksimum elemen yang dapat dipilih (**twoStacks**(**x**, **a**, **n**, **b**, **m**)) ke konsol.