Nama : Khusnia Fitri

NIM : 1203230030

1. Kode:

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

// Definisi struktur node
typedef struct Node {
    char alphabet;
    struct Node* link;
} Node;

// Definisi struktur stack
typedef struct {
    Node* top;
} Stack;

// Fungsi untuk membuat stack baru
Stack* createStack() {
    Stack* stack = (Stack*)malloc(sizeof(Stack));
    stack->top = NULL;
    return stack;
}

// Fungsi untuk mengecek apakah stack kosong
int isEmpty(Stack* stack) {
    return (stack->top == NULL);
}

// Fungsi untuk menambahkan elemen baru ke dalam stack
void push(Stack* stack, char data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->alphabet = data;
    newNode->link = stack->top;
    stack->top = newNode;
}

// Fungsi untuk menghapus dan mengembalikan elemen teratas dari stack
char pop(Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty. Cannot pop.\n");
        return '\0';
    }
```

```
    Node* temp = stack->top;
    char poppedValue = temp->alphabet;
    stack->top = temp->link;
    free(temp);
    return poppedValue;
}

// Fungsi untuk mencetak string "INFORMATIKA" dari stack jika ditemukan
void printInformatika(Node* start) {
    // Definisikan string yang ingin dicari
    char* word = "INFORMATIKA";
    int word_length = 11;

    // Mencari huruf pertama dari string di linked list
    while (start != NULL && start->alphabet != 'I') {
        start = start->link;
    }

    // Jika huruf 'I' tidak ditemukan di linked list
    if (start == NULL) {
        printf("String 'INFORMATIKA' not found.\n");
        return;
    }

    // Membuat stack baru untuk menampung huruf yang ditemukan
    Stack* stack = createStack();

    // Memasukkan huruf-huruf yang ditemukan ke dalam stack
    while (start != NULL) {
        if (start->alphabet == *word) {
            push(stack, start->alphabet);
            word++;
        // Jika seluruh huruf "INFORMATIKA" sudah ditemukan, cetak string
tersebut dari stack
            if (*word == '\0') {
                // Menyimpan huruf-huruf dari stack ke dalam array
                char output[word_length];
                int i = 0;
                while (!isEmpty(stack)) {
                    output[i++] = pop(stack);
                }
          // Mencetak huruf-huruf yang tersimpan dalam array secara terbalik
                for (int j = i - 1; j >= 0; j--) {
                    printf("%c", output[j]);
                }
```

```
                printf("\n");
                return;
            }
        }
        start = start->link;
    }

    // Jika tidak semua huruf "INFORMATIKA" ditemukan dalam linked list
    printf("String 'INFORMATIKA' not found.\n");
}



int main() {
    // Inisialisasi node-node l1 hingga l9
    Node l1, l2, l3, l4, l5, l6, l7, l8, l9;

    // Inisialisasi linked list
    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';

    // Hubungkan node-node sesuai dengan panah arahnya
    l7.link = &l1;
    l1.link = &l8;
    l8.link = &l2;
    l2.link = &l5;
    l5.link = &l3;
    l3.link = &l6;
    l6.link = &l9;
```

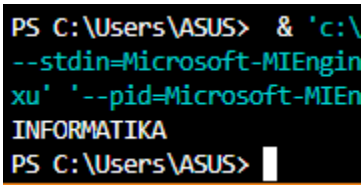
```
    l9.link = &l4;
    l4.link = &l7;

    // Tentukan node awal (starting point) ke l3
    Node* current = &l3;

    // Cetak "INFORMATIKA" jika ditemukan
    printInformatika(current);

    return 0;
}
```

Output :



Penjelasan:

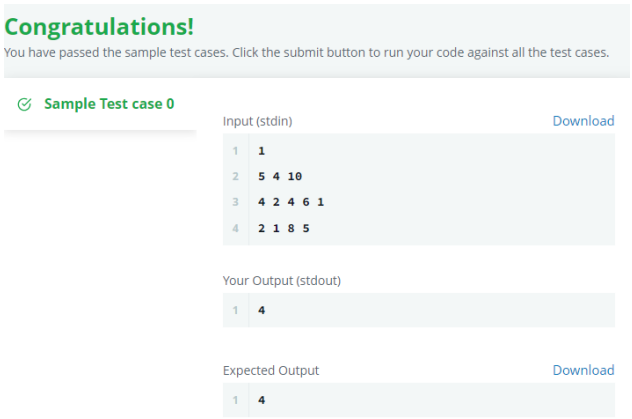
1. **Struktur Node:**
    - Ini adalah struktur data dasar yang digunakan untuk merepresentasikan setiap elemen dalam linked list.
    - Setiap elemen (node) memiliki dua bagian: **alphabe**t: yang menyimpan karakter huruf, **link**: yang adalah pointer ke node berikutnya dalam linked list.
2. **Struktur Stack:**
    - Struktur data yang merepresentasikan tumpukan(stack), digunakan untuk menyimpan karakter huruf dari linked list.
3. **Fungsi createStack():** Fungsi ini membuat stack baru dan mengembalikan pointer ke stack tersebut.
4. **Fungsi isEmpty():** Fungsi ini memeriksa apakah stack kosong. Jika stack kosong, fungsi ini mengembalikan nilai 1; jika tidak, mengembalikan nilai 0.
5. **Fungsi push():** Fungsi ini menambahkan elemen baru ke dalam stack. Elemen yang ditambahkan adalah karakter huruf. Elemen baru ditambahkan di atas (atas stack).
6. **Fungsi pop():** Fungsi ini menghapus dan mengembalikan elemen teratas dari stack. Elemen teratas yang dihapus adalah karakter huruf.
7. **Fungsi printInformatika**():
    - Fungsi ini mencari dan mencetak string "INFORMATIKA" dari linked list.
    - Langkah-langkahnya adalah: Mencari huruf 'I' pertama dalam linked list. Jika huruf 'I' ditemukan, lanjutkan pencarian untuk huruf selanjutnya dari

"INFORMATIKA". Jika seluruh huruf "INFORMATIKA" ditemukan, cetak string tersebut dari stack secara terbalik.

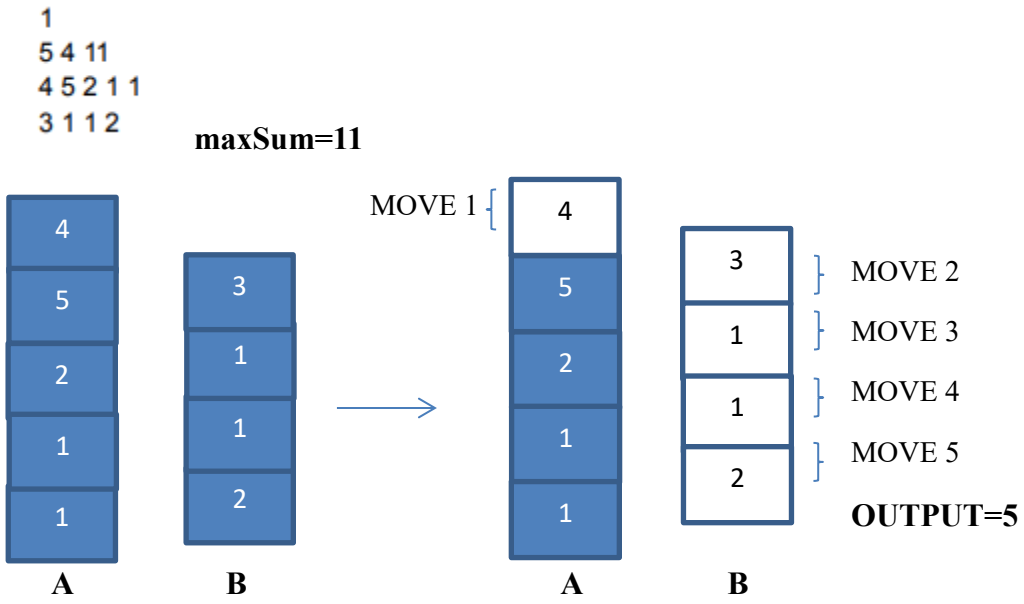
- Jika tidak semua huruf "INFORMATIKA" ditemukan dalam linked list, cetak pesan bahwa string tersebut tidak ditemukan.

8. **Fungsi main():**
    - Membuat linked list dengan karakter huruf yang disediakan. Menghubungkan node-node sesuai dengan arah panahnya.
    - Menentukan node awal (starting point). Memanggil fungsi printInformatika() untuk mencetak string "INFORMATIKA" dari linked list, dimulai dari node awal.

2. Selseaikan https://www.hackerrank.com/challenges/game-of-two-stacks/problem



Tambahkan visualisasi alur penyelesaiannya (per langkah) dalam bentuk stack pada laporan apabila input diubah menjadi sebagai berikut.



Kode :

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main(){
    int g;
    scanf("%d",&g);
    for(int a0 = 0; a0 < g; a0++){
        int n;
        int m;
        int x;
        scanf("%d %d %d",&n,&m,&x);
        int *a = malloc(sizeof(int) * n);
        for(int a_i = 0; a_i < n; a_i++){
           scanf("%d",&a[a_i]);
        }
        int *b = malloc(sizeof(int) * m);
        for(int b_i = 0; b_i < m; b_i++){
           scanf("%d",&b[b_i]);
        }
        // your code goes here
        int max = 0;
        int sum = 0;
        int items = 0;
        int apos = 0, bpos = 0;
        while (sum <= x && apos < n) {
            if (sum + a[apos] > x)
                break;
            sum += a[apos];
            apos++;
            items++;
        }
        while (sum <= x && bpos < m) {
            if (sum + b[bpos] > x)
                break;
            sum += b[bpos];
            bpos++;
            items++;
        }
```

```
        max = items;
        while (1) {
            if (apos <= 0)
                break;
            apos--;
            sum -= a[apos];
            items--;
            while (sum <= x && bpos < m) {
                if (sum + b[bpos] > x)
                    break;
                sum += b[bpos];
                bpos++;
                items++;
            }
            if (items > max)
                max = items;
            if (bpos == m)
                break;
        }
        printf ("%d\n", max);
    }
    return 0;
}
```

Output :

```
1
5 4 11
4 5 2 1 1
3 1 1 2
5
```

Penjelasan :

- Program ini bertujuan untuk menemukan jumlah maksimum elemen yang dapat diambil dari kedua stack sehingga total jumlahnya tidak melebihi **x**
- **int g; scanf("%d",&g);:** Mendeklarasikan variabel **g** yang akan menampung jumlah kasus uji **(g)** yang akan dieksekusi. Menggunakan **fungsi scanf()** untuk membaca nilai **g** dari input).
- **for(int a0 = 0; a0 < g; a0++) :**Looping for yang akan berjalan sebanyak **g** kali, yaitu sebanyak jumlah kasus uji. Di dalam loop, program membaca tiga nilai integer **(n, m, x)**

yang menunjukkan jumlah elemen dalam stack A, jumlah elemen dalam stack B, dan nilai maksimum yang dapat diambil dari kedua stack.

- Nilai-nilai ini dibaca dari input standar menggunakan **scanf(). int *a = malloc(sizeof(int) * n);:** Mengalokasikan memori dinamis untuk array a yang akan menampung elemen-elemen stack A. Memori dialokasikan berdasarkan jumlah elemen (n) yang dibaca sebelumnya.
- **for(int a_i = 0; a_i < n; a_i++) { scanf("%d",&a[a_i]); }:** Looping untuk membaca nilai-nilai elemen stack A dari input standar dan menyimpannya dalam array a.
- Langkah 12-13 dilakukan juga untuk stack B, yaitu **int *b = malloc(sizeof(int) * m);** dan **for(int b_i = 0; b_i < m; b_i++) { scanf("%d",&b[b_i]); }.**
- Setelah membaca elemen-elemen stack A dan B, program mulai mencari kombinasi elemen terbaik yang memenuhi batasan nilai maksimum (x) yang diberikan. Pencarian dilakukan menggunakan beberapa variabel seperti sum, items, apos, dan bpos untuk melacak jumlah elemen yang telah diambil dari setiap stack dan posisi saat ini dalam setiap stack. Setelah menemukan jumlah maksimum elemen yang dapat diambil, program mencetak jumlah tersebut menggunakan printf(). Looping for utama berakhir setelah semua kasus uji selesai dieksekusi. Fungsi main() mengembalikan nilai 0, menandakan bahwa program telah berakhir dengan sukses.