NAMA: SATRIO DWI YANDA ARIFIN

NIM: 1203230020

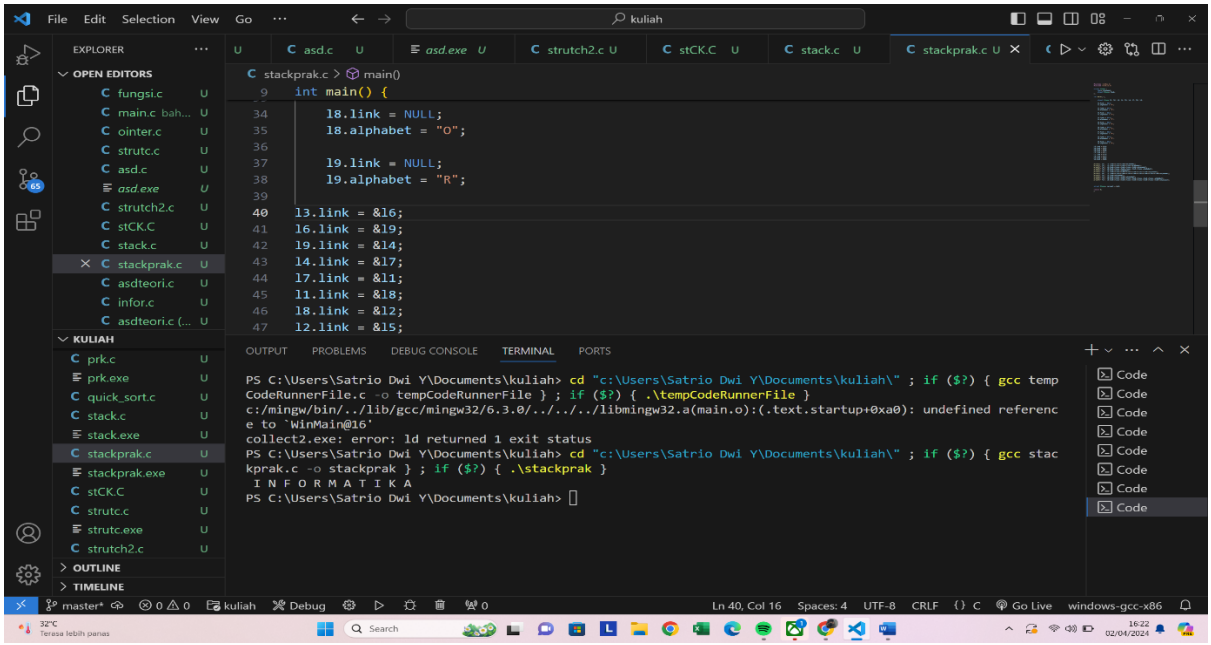
KELAS: IF 03-01

SOURCE 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";
```

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


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



struct Stone* current = &l3;

return 0;
}
```

HASIL OUTPUT



PENJELASAN

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

Mendefinisikan struktur **`Stone`** yang memiliki dua anggota: **`alphabet`** bertipe pointer ke karakter (digunakan untuk menyimpan huruf alfabet) dan **`link`** bertipe pointer ke struktur **`Stone`** (digunakan untuk membuat linked list).

```
int main() {
    // Deklarasi variabel-variabel batu
    struct Stone l1, l2, l3, l4, l5, l6, l7, l8, l9;

    // Inisialisasi batu-batu dengan huruf alfabet
    l1.link = NULL;
    l1.alphabet = "F";
    // ...
    l9.link = NULL;
    l9.alphabet = "R";

    // Menghubungkan batu-batu untuk membuat linked list
    l3.link = &l6;
    l6.link = &l9;
    // ...

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

    // Penunjuk saat ini
    struct Stone* current = &l3;

    return 0;
}
```

1.

Program dimulai dengan deklarasi dari batu-batu yang diberi nama **`l1`** hingga **`l9`**. Setiap batu diinisialisasi dengan **`link`** yang menunjuk ke **`NULL`** (akhir dari linked list) dan **`alphabet`** yang berisi huruf alfabet sesuai urutan.

Batu-batu kemudian dihubungkan satu sama lain dengan menggunakan operator `->link`.
Setelah itu, program mencetak beberapa huruf alfabet dari linked list dengan mengakses pointer `link` secara berurutan.
Terakhir, sebuah pointer `current` diinisialisasi untuk menunjuk pada batu ketiga (`l3`).

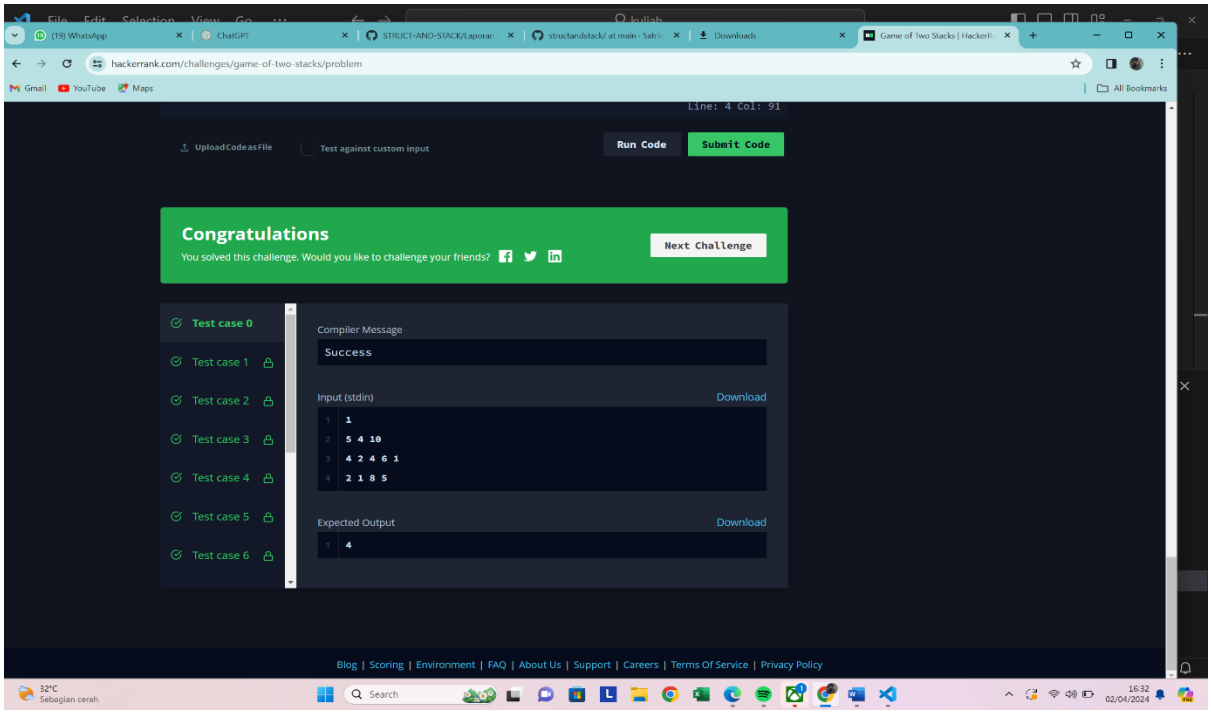
Dengan cara ini, program membuat dan mengelola sebuah linked list sederhana yang berisi huruf alfabet, dan kemudian mencetak sebagian dari linked list tersebut.

SOAL 2

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

int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
    int sum = 0;
    int count = 0;
    int maxCount = 0;
    int i = 0, j = 0;

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

    maxCount = count;

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

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

        if (sum <= maxSum && i + j > maxCount) {
            maxCount = i + j;
        }
    }

    return maxCount;
}

int main() {
    int g;
    scanf("%d", &g);

    for (int g_itr = 0; g_itr < g; g_itr++) {

        int n, m, maxSum;
        scanf("%d %d %d", &n, &m, &maxSum);

        int* a = malloc(n * sizeof(int));
        for (int i = 0; i < n; i++) {
```

```
            scanf("%d", &a[i]);
        }

        int* b = malloc(m * sizeof(int));
        for (int i = 0; i < m; i++) {
            scanf("%d", &b[i]);
        }

        int result = twoStacks(maxSum, n, a, m, b);

        printf("%d\n", result);

        free(a);
        free(b);
    }

    return 0;
}
```

HASIL OUTPUT



PENJELASAN

```
int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
    int sum = 0;
```

```
    int count = 0;
    int maxCount = 0;
    int i = 0, j = 0;

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

    maxCount = count;

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

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

        if (sum <= maxSum && i + j > maxCount) {
            maxCount = i + j;
        }
    }

    return maxCount;
}
```

Fungsi **`twoStacks`** merupakan inti dari program ini. Fungsi ini menerima lima parameter:
`maxSum`: Batas jumlah maksimum yang tidak boleh dilewati.
`a_count`: Jumlah elemen dalam tumpukan pertama.
`a`: Pointer ke array yang berisi elemen-elemen tumpukan pertama.
`b_count`: Jumlah elemen dalam tumpukan kedua.
`b`: Pointer ke array yang berisi elemen-elemen tumpukan kedua.
Fungsi ini mengembalikan jumlah maksimum elemen yang dapat diambil dari kedua tumpukan sehingga jumlah total elemen yang diambil tidak melebihi **`maxSum`**.
Algoritma yang digunakan di sini menghitung jumlah maksimum elemen dengan dua pointer **`i`** dan **`j`** yang melacak elemen yang diambil dari kedua tumpukan.
Fungsi ini menggunakan dua buah loop **`while`** untuk mengecek kondisi dan melakukan perhitungan sesuai dengan algoritma yang telah dijelaskan.

```
int main() {
    int g;
    scanf("%d", &g);

    for (int g_itr = 0; g_itr < g; g_itr++) {
```

```
        int n, m, maxSum;
        scanf("%d %d %d", &n, &m, &maxSum);

        int* a = malloc(n * sizeof(int));
        for (int i = 0; i < n; i++) {
            scanf("%d", &a[i]);
        }

        int* b = malloc(m * sizeof(int));
        for (int i = 0; i < m; i++) {
            scanf("%d", &b[i]);
        }

        int result = twoStacks(maxSum, n, a, m, b);

        printf("%d\n", result);

        free(a);
        free(b);
    }

    return 0;
}
```

Fungsi `main` adalah titik masuk utama program.Pertama-tama, fungsi ini membaca jumlah kasus uji (`g`) menggunakan fungsi `scanf`.Selanjutnya, dilakukan perulangan `for` untuk setiap kasus ujiDalam setiap iterasi perulangan, program membaca nilai `n`, `m`, dan `maxSum` menggunakan `scanf`.Memori dialokasikan untuk array `a` dan `b` menggunakan `malloc` sesuai dengan jumlah elemen yang dibaca.Nilai-nilai elemen untuk tumpukan pertama dan kedua dibaca menggunakan `scanf`.Fungsi `twoStacks` dipanggil dengan parameter yang sesuaiHasilnya dicetak menggunakan `printf`.Setelah semua kasus uji selesai dieksekusi, memori yang dialokasikan untuk array `a` dan `b` dibebaskan menggunakan `free`.Fungsi `main` mengembalikan nilai `0`, menandakan bahwa program berakhir dengan sukses.