

Nama : Khansa Nailah Anjani

Nim : 1203230038

Kelas : IF-03-02

SOAL NOMOR 1

-SOURCE CODE

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

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

int main() {
    // Mendeklarasi node-node
    struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
    struct Node *link, *l3ptr;

    // Inisialisasi node-node dengan menggunakan potongan kode soal
    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;
```

```

19.alphabet = "R";

// Mengatur koneksi antar node sesuai dengan urutan yang diinginkan
17.link = &l1; // Menyambungkan ke l1
11.link = &l8; // Menyambungkan ke l1
18.link = &l2; // Menyambungkan ke l1
12.link = &l5; // Menyambungkan ke l1
15.link = &l3; // Menyambungkan ke l1
13.link = &l6; // Menyambungkan ke l1
16.link = &l9;
19.link = &l4;
14.link = &l7;

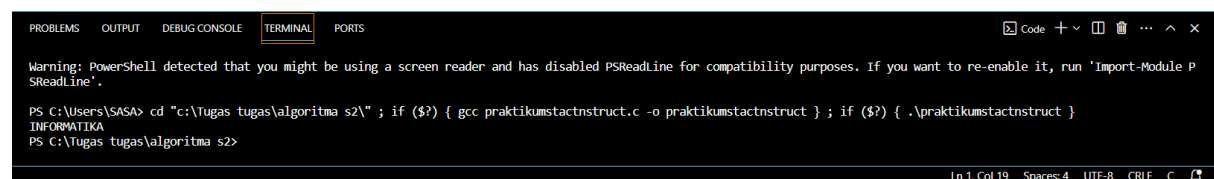
// Starting point
l3ptr = &l7;

// Akses data menggunakan printf dengan syarat struktur code yang disuruh
printf("%s", l3.link->link->link->alphabet); // Menampilkan huruf I
printf("%s", l3.link->link->link->link->alphabet); // Menampilkan huruf N
printf("%s", l3.link->link->link->link->link->alphabet); // Menampilkan
huruf F
printf("%s", l3.link->link->link->link->link->link->alphabet); //
Menampilkan huruf O
printf("%s", l3.link->link->alphabet); // Menampilkan huruf R
printf("%s", l3.link->link->link->link->link->link->link->alphabet); //
Menampilkan huruf M
printf("%s", l3.alphabet); // Menampilkan huruf A
printf("%s", l3.link->alphabet); // Menampilkan huruf T
printf("%s", l3.link->link->link->alphabet); // Menampilkan huruf I
printf("%s", l3.link->link->link->link->link->link->link->link-
>alphabet); // Menampilkan huruf K
printf("%s", l3.alphabet); // Menampilkan huruf A

return 0;
}

```

-SS OUTPUT



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\SASA> cd "c:\Tugas tugas\algoritma s2\" ; if ($?) { gcc praktikumstactnstruct.c -o praktikumstactnstruct } ; if ($?) { .\praktikumstactnstruct }
INFORMATIKA
PS C:\Tugas tugas\algoritma s2>

```

Ln 1, Col 19 Spaces: 4 UTF-8 CRLF C

-PENJELASAN CODE PROGRAM

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

Baris diatas menyertakan file header `stdio.h` yang berisi definisi fungsi standar untuk input dan output seperti `printf` dan `scanf` dan menyertakan file header `stdlib.h` yang berisi definisi fungsi standar untuk memori seperti `malloc` dan `free`.

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

Mendeklarasikan struktur data baru bernama `Node`. Mendeklarasikan anggota struktur `Node` bernama `alphabet` yang bertipe pointer ke karakter (string). Mendeklarasikan anggota struktur `Node` bernama `link` yang bertipe pointer ke struktur `Node`. Jadi dia akan mendefinisikan struktur data `Node` dengan dua anggota: `alphabet` dan `link`. Kemudian akan memungkinkan program untuk membuat, menghubungkan, dan mengakses node dalam linked list.

```
int main() {
    // Mendeklarasi node-node
    struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
    struct Node *link, *l3ptr;

    // Inisialisasi node-node dengan menggunakan potongan kode soal
    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";
```

```

18.link = NULL;
18.alphabet = "O";

19.link = NULL;
19.alphabet = "R";

```

Disini merupakan fungsi utama dari program ini, Dimana struct Node 11, 12, 13, 14, 15, 16, 17, 18, 19; Mendeklarasikan 9 variabel bertipe struct Node dengan nama 11 hingga 19. Dan struct Node *link, *l3ptr; Mendeklarasikan 2 pointer bertipe struct Node dengan nama link dan l3ptr. Kemudian melakukan inisialisasi nilai awal untuk setiap node: link: Diinisialisasi dengan NULL untuk menunjukkan bahwa node tidak terhubung ke node lain. alphabet: Diinisialisasi dengan huruf yang sesuai (misalnya, 11.alphabet diinisialisasi dengan "F") dan kode program lainnya yang menggunakan node-node yang telah dideklarasikan dan diinisialisasi. Sempelnya baris kode di atas mendeklarasikan variabel, pointer, dan menginisialisasi node-node yang akan digunakan dalam linked list. Alur program menunjukkan langkah-langkah eksekusi program, dimulai dari deklarasi, inisialisasi, dan eksekusi kode program.

```

// Mengatur koneksi antar node sesuai dengan urutan yang diinginkan
17.link = &l1; // Menyambungkan ke 11
11.link = &l8; // Menyambungkan ke 11
18.link = &l2; // Menyambungkan ke 11
12.link = &l5; // Menyambungkan ke 11
15.link = &l3; // Menyambungkan ke 11
13.link = &l6; // Menyambungkan ke 11
16.link = &l9;
19.link = &l4;
14.link = &l7;

```

Baris-baris kode di atas mengatur koneksi antar node dalam linked list sesuai dengan urutan yang diinginkan. Kode ini menggunakan pointer link untuk menghubungkan setiap node ke node berikutnya. Pertama: 17.link dihubungkan ke &l1. Ini berarti node 17 akan menunjuk ke node 11. Selanjutnya: 11.link dihubungkan ke &l8. Ini berarti node 11 akan menunjuk ke node 18. Proses ini diulang untuk node-node lainnya, menghubungkan setiap node ke node berikutnya dalam urutan yang diinginkan. Hasilnya: Sebuah linked list terbentuk dengan urutan node: 17 -> 11 -> 18 -> 12 -> 15 -> 13 -> 16 -> 19 -> 14 -> 17.

```

// Starting point
l3ptr = &l7;

```

Baris kode ini menetapkan nilai &l7 ke pointer l3ptr. Pertama kita melakukan penetapan nilai: Pointer l3ptr diinisialisasi dengan alamat memori dari node 17 kemudian kita melakukan starting point: Node 17 menjadi titik awal untuk menelusuri linked list.

```

// Akses data menggunakan printf
printf("%s", l3.link->link->link->alphabet); // Menampilkan huruf I
printf("%s", l3.link->link->link->link->alphabet); // Menampilkan huruf N
printf("%s", l3.link->link->link->link->link->alphabet); // Menampilkan
huruf F
printf("%s", l3.link->link->link->link->link->link->alphabet); //
Menampilkan huruf O
printf("%s", l3.link->link->alphabet); // Menampilkan huruf R
printf("%s", l3.link->link->link->link->link->link->link->alphabet); //
Menampilkan huruf M
printf("%s", l3.alphabet); // Menampilkan huruf A
printf("%s", l3.link->alphabet); // Menampilkan huruf T
printf("%s", l3.link->link->link->alphabet); // Menampilkan huruf I
printf("%s", l3.link->link->link->link->link->link->link->link-
>alphabet); // Menampilkan huruf K
printf("%s", l3.alphabet); // Menampilkan huruf A

return 0;
}

```

Baris-baris kode di atas menggunakan fungsi printf untuk mencetak data (huruf) pada node-node dalam linked list. Untuk alurnya kurang lebih seperti ini pertama kita melakukan pengaksesan node: Kode menggunakan l3.link untuk mengakses node berikutnya setelah l3. Selanjutnya terjadi proses pengulangan: Kode menggunakan link dari node yang diakses untuk mengakses node berikutnya dan seterusnya. Kemudian yang terakhir pencetakan: Kode menggunakan printf untuk mencetak huruf (data alphabet) pada node yang diakses. dan program telah selesai.

SOAL NOMOR 2

-SOURCE CODE

```

#include <stdio.h>

int twoStacks(int maxSum, int a[], int n, int b[], int m) {
    int sum = 0, count = 0, temp = 0, i = 0, j = 0;

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

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

```

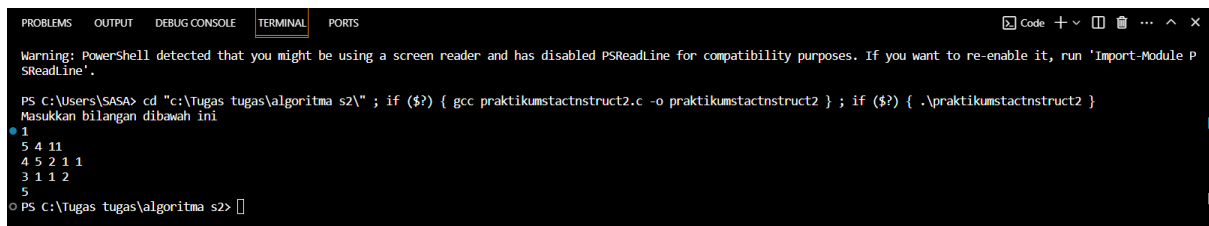
```

        count = i + j;
    }
}
return count;
}

int main() {
    int g;
    printf("Masukkan bilangan dibawah ini \n");
    scanf("%d", &g);
    while (g--) {
        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]);
        }
        printf("%d\n", twoStacks(maxSum, a, n, b, m));
    }
    return 0;
}

```

-SS INPUT DAN OUTPUT



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadline for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadline'.
PS C:\Users\SASA> cd "c:\Tugas tugas\algoritma s2\"; if ($?) { gcc praktikumstactnstruct2.c -o praktikumstactnstruct2 }; if ($?) { .\praktikumstactnstruct2 }
Masukkan bilangan dibawah ini
1
5 4 11
4 5 2 1 1
3 1 1 2
5
o PS C:\Tugas tugas\algoritma s2>

```

-PENJELASAN CODE PROGRAM

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

pada program ini kita menggunakan satu header file saja yaitu stdio.h yang berisi definisi fungsi standar untuk input dan output seperti printf dan scanf.

```

int twoStacks(int maxSum, int a[], int n, int b[], int m) {
    int sum = 0, count = 0, temp = 0, i = 0, j = 0;

```

baris ini menginisialisasi: Variabel sum, count, temp, i, dan j diinisialisasi. Dan kode lainnya bekerja untuk mengiterasi array a dan b menggunakan i dan j. melakukan perbandingan elemen dari a dan b. selanjutnya menambahkan elemen ke "stack" (mungkin diimplementasikan dengan struktur data lain) jika memenuhi kondisi tertentu, seperti: total

penjumlahan tidak melebihi maxSum. elemen memenuhi kriteria tertentu (tergantung implementasi). Kemudian meningkatkan count jika elemen ditambahkan ke stack. melakukan perhitungan dan operasi lain (menggunakan temp). dan yang terakhir pengembalian: Fungsi twoStacks mengembalikan nilai int (biasanya count atau nilai lain yang relevan).

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

Baris kode ini melakukan perulangan untuk menambahkan elemen array a ke sum selama kondisinya terpenuhi. Nilai akhir i (yang menunjukkan indeks terakhir yang diproses) disimpan ke variabel count.

```
while (j < m && i >= 0) {  
    sum += b[j++];  
    while (sum > maxSum && i > 0) {  
        sum -= a[--i];  
    }  
    if (sum <= maxSum && i + j > count) {  
        count = i + j;  
    }  
}  
return count;  
}
```

Baris kode ini akan mengimplementasikan algoritma untuk menemukan jumlah maksimum elemen yang dapat ditambahkan dari dua array a dan b ke dalam stack, dengan batasan total penjumlahan maxSum. Algoritma ini melakukan iterasi pada kedua array dan menambahkan elemen secara bergantian, sambil memastikan total penjumlahan tidak melebihi maxSum.

```
int main() {  
    int g;  
    printf("Masukkan bilangan dibawah ini \n");  
    scanf("%d", &g);  
    while (g--) {  
        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]);  
        }  
        printf("%d\n", twoStacks(maxSum, a, n, b, m));  
    }  
}
```

```
    return 0;  
}
```

Disini adalah code utama dari program ini Dimana pertama akan mendeklarasikan variable g untuk menyimpan jumlah inputan user kemudian program akan meminta user untuk menginputkan angka, setelah itu disini kita menggunakan perulangan while dan akan berjalan sebanyak g kali. didalam while si program akan membaca nilai n, m dan maxsum dari user kemudian terdapat 2 array yaitu a dan b Dimana masing masing mendeklarasikan Panjang n dan m masing masing, kemudian terdapat perulangan for untuk membaca elemen dari array a dan b kemudian akan mencetak hasil dari fungsi twostacks dan yang terakhir akan melakukan pengembalian nilai 0.