

Muh Richard Daneswara

1203230046

IF 03-02

Source code

Nomor 1

```
#include <stdio.h>

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

int main()
{
    struct node
    11, 12,
    13, 14,
    15, 16,
    17, 18,
    19;

    11.link = NULL;
    11.alphabet = 'F';

    12.link = NULL;
    12.alphabet = 'M';

    13.link = NULL;
    13.alphabet = 'A';

    14.link = NULL;
    14.alphabet = 'I';

    15.link = NULL;
    15.alphabet = 'K';

    16.link = NULL;
    16.alphabet = 'T';

    17.link = NULL;
    17.alphabet = 'N';

    18.link = NULL;
```

```

18.alphabet = 'O';

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

14.link = &17; 17.link = &11;
11.link = &18; 18.link = &19;
19.link = &12; 12.link = &13;
13.link = &16; 16.link = &14;

printf("%c", 14.alphabet); //I//
printf("%c", 14.link->alphabet); //N//
printf("%c", 14.link->link->alphabet); //F//
printf("%c", 14.link->link->link-
>alphabet); //O//
printf("%c", 14.link->link->link->link->alphabet); //R//
printf("%c", 14.link->link->link->link->link->alphabet); //M//
printf("%c", 14.link->link->link->link->link->link-
>alphabet); //A//
printf("%c", 14.link->link->link->link->link->link->link-
>alphabet); //T//
printf("%c", 14.link->link->link->link->link->link->link->link-
>alphabet); //I//

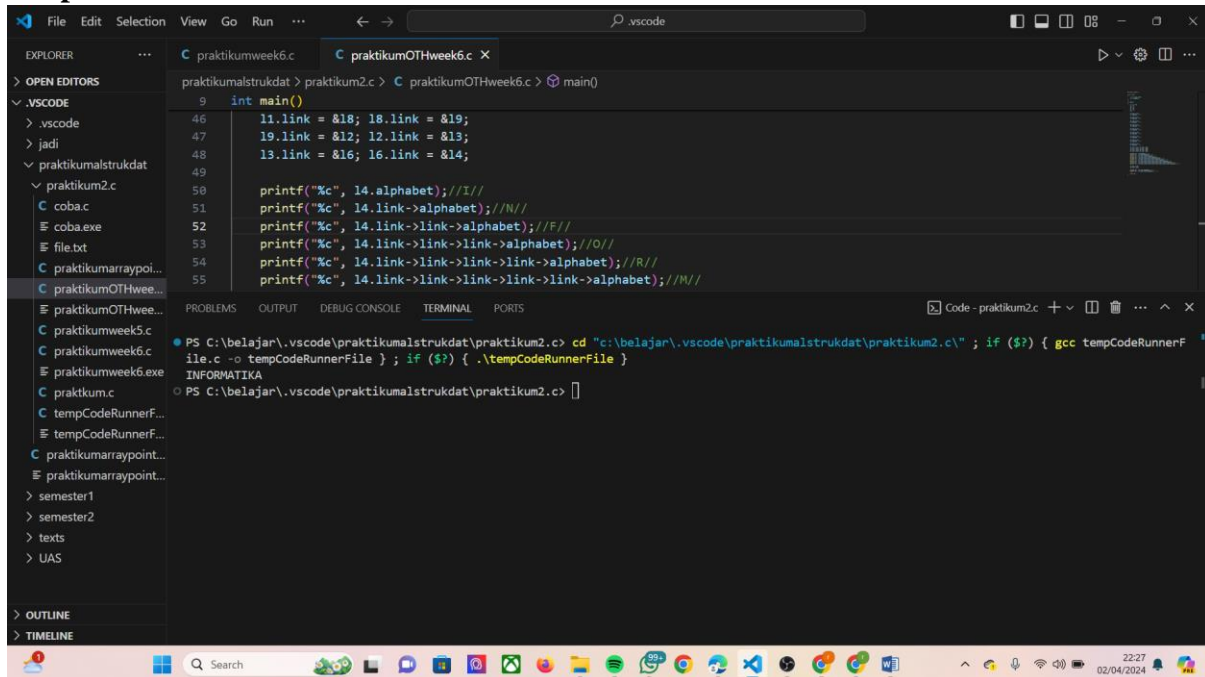
14.link = &15;
15.link = &13;

printf("%c", 14.link->alphabet); // K //
printf("%c", 14.link->link->alphabet); // A //

return 0;
}

```

Output



```
praktikumalstruktat > praktikum2.c > C praktikumOTHweek6.c > main()
9  int main()
46  11.link = &18; 18.link = &19;
47  19.link = &12; 12.link = &13;
48  13.link = &16; 16.link = &14;
49
50  printf("%c", 14.alphabet); //I//
51  printf("%c", 14.link->alphabet); //N//
52  printf("%c", 14.link->link->alphabet); //E//
53  printf("%c", 14.link->link->link->alphabet); //O//
54  printf("%c", 14.link->link->link->link->alphabet); //R//
55  printf("%c", 14.link->link->link->link->link->alphabet); //M//

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\belajar\.vscode\praktikumalstruktat\praktikum2.c> cd "c:\belajar\.vscode\praktikumalstruktat\praktikum2.c\"; if ($?) { gcc tempCodeRunnerF
ile.c -o tempCodeRunnerFile }; if ($?) { .\tempCodeRunnerFile }
INFORMATIKA
PS C:\belajar\.vscode\praktikumalstruktat\praktikum2.c> []
```

Penjelasan

struct node

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

Baris ini mendefinisikan sebuah struktur yang disebut node. Struktur ini memiliki dua anggota: link yang merupakan pointer ke struktur node berikutnya, dan alphabet yang merupakan karakter.

int main()

```
{
```

Fungsi utama dimulai di sini.

```
    struct node 11, 12, 13, 14, 15, 16, 17, 18, 19;
```

Baris ini mendeklarasikan sembilan variabel bertipe struct node, yaitu 11 sampai 19, yang akan digunakan sebagai simpul dalam linked list.

```
    11.link = NULL;
```

```
    11.alphabet = 'F';
```

Setiap variabel node diinisialisasi secara individu di sini. Setiap variabel diberi nilai untuk kedua anggotanya, yaitu link yang diatur menjadi NULL (awalnya tidak terhubung dengan simpul mana pun) dan alphabet yang diisi dengan karakter tertentu.

```
l4.link = &l7; // N//
```

Baris-baris ini menghubungkan simpul-simpul satu sama lain untuk membentuk linked list. Misalnya, `l4.link = &l7` menghubungkan simpul 14 ke simpul 17.

```
printf("%c", l4.alphabet); // I//  
printf("%c", l4.link->alphabet); // N//
```

Baris-baris ini mencetak isi linked list yang telah dibentuk sebelumnya. Ini menggunakan pointer untuk mengakses anggota dari simpul-simpul yang terhubung satu sama lain.

```
l4.link = &l5;  
l5.link = &l3;
```

Baris-baris ini mengubah hubungan antar simpul dengan mengubah nilai pointer link dari beberapa simpul. Ini mengubah struktur linked list.

```
printf("%c", l4.link->alphabet);    // K//  
printf("%c", l4.link->link->alphabet); // A//
```

Baris-baris ini mencetak isi linked list setelah perubahan yang dilakukan sebelumnya. Hal ini memperlihatkan dampak dari perubahan terhadap struktur linked list.

```
return 0;  
}
```

Fungsi main selesai di sini. Ini adalah akhir dari program, dan nilai 0 dikembalikan, menunjukkan bahwa program berjalan dengan sukses.

Nomor 2

```
#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;
    scanf("%d", &g);
    while (g-- > 0) {
        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;
}

```

Output

The screenshot shows the Visual Studio Code editor with a C program open. The program defines a function `twoStacks` that takes a maximum sum and two arrays, and returns the number of ways to reach the maximum sum by adding elements from the arrays. The `main` function reads the number of test cases `g`, and for each case, it reads `n`, `m`, and `maxSum`, then reads the arrays `a` and `b`, and prints the result of `twoStacks`.

The terminal output shows the execution of the program. The first test case has `n=5`, `m=4`, and `maxSum=10`. The arrays are `a = [4, 2, 4, 6, 1]` and `b = [2, 1, 8, 5]`. The output is `4`, indicating there are 4 ways to reach the maximum sum of 10.

```

PS C:\belajar\.vscode\praktikumalstrukdat\praktikum2.c> cd "c:\belajar\.vscode\praktikumalstrukdat\praktikum2.c\"; if ($?) { gcc praktikumOTHwe
ek6.c -o praktikumOTHweek6 }; if ($?) { .\praktikumOTHweek6 }
5 4 10
4 2 4 6 1
2 1 8 5
4
PS C:\belajar\.vscode\praktikumalstrukdat\praktikum2.c>

```

Penjelasan

```
int twoStacks(int maxSum, int a[], int n, int b[], int m) {
```

Ini adalah definisi fungsi `twoStacks()`. Fungsi ini mengambil beberapa parameter, yaitu `maxSum` (jumlah maksimum yang bisa diambil), array `a` dengan ukuran `n` dan array `b` dengan ukuran `m`. Fungsi ini diharapkan mengembalikan jumlah maksimum elemen yang bisa diambil dari dua stack sedemikian hingga total nilai elemen tidak melebihi `maxSum`.

```
    int sum = 0, count = 0, temp = 0, i = 0, j = 0;
```

Variabel-variabel lokal yang digunakan dalam fungsi `twoStacks()`. Variabel `sum` digunakan untuk menyimpan jumlah elemen yang telah diambil, `count` menyimpan jumlah elemen terbesar yang telah diambil, `temp` digunakan sementara dalam penghitungan, dan `i` dan `j` adalah indeks yang digunakan untuk traversal array `a` dan `b`.

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

Loop pertama untuk menambahkan elemen-elemen dari array `a` ke dalam stack pertama (`sum`). Loop ini akan berhenti ketika jumlah `sum` melebihi `maxSum` atau semua elemen `a` telah diambil.

```
    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;  
        }  
    }
```

Loop kedua untuk menambahkan elemen-elemen dari array `b` ke dalam stack kedua (`sum`). Selama `sum` melebihi `maxSum`, loop ini akan mengurangi elemen-elemen dari stack pertama hingga `sum` kembali kurang dari atau sama dengan `maxSum`. Setiap kali jumlah elemen dari kedua stack lebih besar dari `count`, `count` akan diperbarui dengan jumlah tersebut.

```
    return count;  
}
```

Fungsi `twoStacks()` mengembalikan nilai `count` setelah loop-loop selesai dieksekusi.

```
int main() {
```

Fungsi `main()` dimulai di sini.

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

Variabel g yang menyimpan jumlah kasus uji dibaca dari input standar menggunakan fungsi scanf().

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

Loop while ini akan dieksekusi sebanyak g kali, masing-masing kali menangani satu kasus uji.

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

Tiga nilai, yaitu n, m, dan maxSum, dibaca dari input standar menggunakan fungsi scanf().

```
int a[n], b[m];
```

Array a dan b dideklarasikan dengan ukuran yang telah dibaca sebelumnya.

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

Loop untuk membaca elemen-elemen array a dari input standar.

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

Loop untuk membaca elemen-elemen array b dari input standar.

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

Setelah semua input untuk satu kasus uji telah dibaca dan disimpan dalam array, fungsi twoStacks() dipanggil untuk menemukan jumlah maksimum elemen yang bisa diambil dari dua stack dengan batasan maxSum, dan hasilnya dicetak ke output standar.

```
return 0;  
}
```

Fungsi main() diakhiri, dan program selesai. Ini mengembalikan nilai 0, menandakan bahwa program telah berjalan dengan sukses.