

NAMA : Yonathan Hari Dharmawan

NIM : 1203230050

KELAS: IF 03-03

Tugas 1

1. Source Code

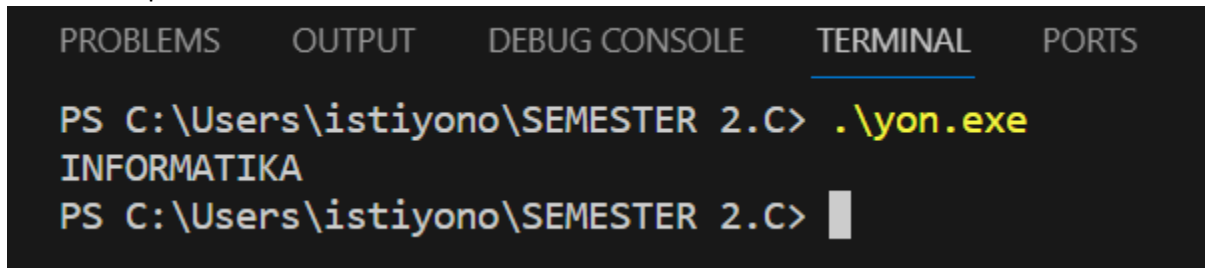
```
2. #include <stdio.h>
3.
4. struct node
5. {
6.     struct node *link;
7.     char alphabet;
8. };
9.
10. int main()
11. {
12.     // Node initialization
13.     struct node l1, l2, l3, l4, l5, l6, l7, l8, l9;
14.
15.     l1.link = NULL;
16.     l1.alphabet = 'F';
17.
18.     l2.link = NULL;
19.     l2.alphabet = 'M';
20.
21.     l3.link = NULL;
22.     l3.alphabet = 'A';
23.
24.     l4.link = NULL;
25.     l4.alphabet = 'I';
26.
27.     l5.link = NULL;
28.     l5.alphabet = 'K';
29.
30.     l6.link = NULL;
31.     l6.alphabet = 'T';
32.
33.     l7.link = NULL;
34.     l7.alphabet = 'N';
35.
```

```

36.     18.link = NULL;
37.     18.alphabet = 'O';
38.
39.     19.link = NULL;
40.     19.alphabet = 'R';
41.
42.     // Linking nodes
43.     14.link = &17; // N
44.     17.link = &11; // F
45.     11.link = &18; // O
46.     18.link = &19; // R
47.     19.link = &12; // M
48.     12.link = &13; // A
49.     13.link = &16; // T
50.     16.link = &14; // I
51.
52.     // Print linked list
53.     printf("%c",
14.alphabet); // I
54.     printf("%c", 14.link->
>alphabet); // N
55.     printf("%c", 14.link->link->
>alphabet); // F
56.     printf("%c", 14.link->link->link->
>alphabet); // O
57.     printf("%c", 14.link->link->link->link->
>alphabet); // R
58.     printf("%c", 14.link->link->link->link->link->
>alphabet); // M
59.     printf("%c", 14.link->link->link->link->link->link->
>alphabet); // A
60.     printf("%c", 14.link->link->link->link->link->link->link->
>alphabet); // T
61.     printf("%c", 14.link->link->link->link->link->link->link->link->
>alphabet); // I
62.
63.     14.link = &15;
64.     15.link = &13;
65.
66.     printf("%c", 14.link->alphabet); // K
67.     printf("%c", 14.link->link->alphabet); // A
68.
69.     return 0;
70. }
71.

```

72. Output



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

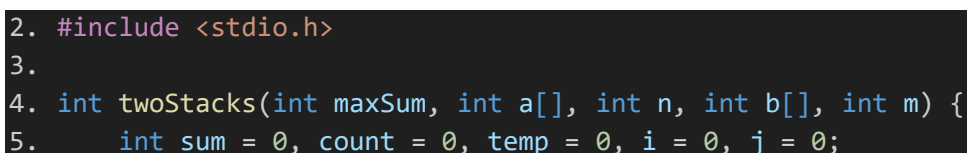
PS C:\Users\istiyono\SEMESTER 2.C> .\yon.exe
INFORMATIKA
PS C:\Users\istiyono\SEMESTER 2.C> 
```

Penjelasan Code

1. `#include <stdio.h>`: Menggunakan preprocessor directive `#include` untuk menyertakan file header `stdio.h`, yang diperlukan untuk fungsi input-output standar.
2. Mendefinisikan struktur node yang berisi pointer ke struktur node itu sendiri (link) dan karakter alphabet.
3. `int main()`: Memulai fungsi utama.
4. Mendeklarasikan beberapa variabel struct node sebagai node-node linked list.
5. Menginisialisasi setiap node dengan NULL untuk link dan karakter yang sesuai untuk alphabet.
6. Melakukan penghubungan antar node dengan menetapkan alamat node berikutnya ke pointer link masing-masing node.
7. Mencetak isi linked list mulai dari node l4 dengan menggunakan pointer link untuk mengakses node berikutnya dan mencetak karakter alphabet masing-masing node.
8. `l4.link = &l5`; Mengubah pointer link dari node l4 untuk menunjuk ke node l5.
9. `l5.link = &l3`; Mengubah pointer link dari node l5 untuk menunjuk ke node l3.
10. Mencetak isi linked list mulai dari node l4 setelah perubahan, seperti yang dilakukan sebelumnya.
11. `return 0`; Mengembalikan nilai 0, menandakan bahwa program telah berakhir dengan sukses.

Tugas 2

1. Source Code



```
2. #include <stdio.h>
3.
4. int twoStacks(int maxSum, int a[], int n, int b[], int m) {
5.     int sum = 0, count = 0, temp = 0, i = 0, j = 0;
```

```

6.
7.     while (i < n && sum + a[i] <= maxSum) {
8.         sum += a[i++];
9.     }
10.    count = i;
11.
12.    while (j < m && i >= 0) {
13.        sum += b[j++];
14.        while (sum > maxSum && i > 0) {
15.            sum -= a[--i];
16.        }
17.        if (sum <= maxSum && i + j > count) {
18.            count = i + j;
19.        }
20.    }
21.    return count;
22.}
23.
24.int main() {
25.    int g;
26.    scanf("%d", &g);
27.    while (g--) {
28.        int n, m, maxSum;
29.        scanf("%d%d%d", &n, &m, &maxSum);
30.        int a[n], b[m];
31.        for (int i = 0; i < n; i++) {
32.            scanf("%d", &a[i]);
33.        }
34.        for (int i = 0; i < m; i++) {
35.            scanf("%d", &b[i]);
36.        }
37.        printf("%d\n", twoStacks(maxSum, a, n, b, m));
38.    }
39.    return 0;
40.}

```

2. Output

Congratulations

You solved this challenge. Would you like to challenge your friends?

[Next Challenge](#)

✓ Test case 0

✓ Test case 1

✓ Test case 2

✓ Test case 3

✓ Test case 4

✓ Test case 5

✓ Test case 6

Compiler Message

Success

Input (stdin)

[Download](#)

```
1 1
2 5 4 10
3 4 2 4 6 1
4 2 1 8 5
```

Expected Output

[Download](#)

```
1 4
```

Penjelasan Code

1. `#include <stdio.h>`: Menggunakan preprocessor directive `#include` untuk menyertakan file header `stdio.h`, yang diperlukan untuk fungsi input-output standar.
2. `int twoStacks(int maxSum, int a[], int n, int b[], int m) {`: Mendefinisikan fungsi `twoStacks` yang mengambil argumen berupa nilai maksimum (`maxSum`), dua array `a` dan `b`, serta ukuran masing-masing array (`n` dan `m`).
3. Deklarasi variabel lokal `sum`, `count`, `temp`, `i`, dan `j` yang akan digunakan dalam fungsi.
4. `while (i < n && sum + a[i] <= maxSum) { ... }`: Menggunakan loop `while` untuk menghitung jumlah elemen dari stack pertama (`a`) yang dapat diambil sehingga jumlah totalnya tidak melebihi `maxSum`.
5. `count = i;`: Menyimpan jumlah elemen yang dapat diambil dari stack pertama pada variabel `count`.
6. `while (j < m && i >= 0) { ... }`: Menggunakan loop `while` untuk mengambil elemen dari stack kedua (`b`) dan mencari kombinasi jumlah maksimum elemen yang dapat diambil dari kedua stack.
7. `sum += b[j++];`: Menambahkan nilai elemen dari stack kedua ke dalam variabel `sum`, dan kemudian menaikkan nilai indeks `j`.

8. `while (sum > maxSum && i > 0) { ... }`: Menggunakan loop while untuk mengurangi elemen dari stack pertama sampai jumlah total elemen dari kedua stack tidak melebihi maxSum.
9. `if (sum <= maxSum && i + j > count) { ... }`: Memeriksa apakah jumlah elemen yang dapat diambil dari kedua stack lebih besar dari nilai count yang sebelumnya disimpan.
10. `count = i + j;`: Memperbarui nilai count dengan jumlah elemen maksimum yang dapat diambil dari kedua stack.
11. `return count;`: Mengembalikan jumlah maksimum elemen yang dapat diambil dari kedua stack.
12. `int main() {`: Memulai fungsi utama.
13. Deklarasi variabel lokal g untuk menyimpan jumlah kasus uji yang akan dijalankan.
14. `scanf("%d", &g);`: Menggunakan fungsi scanf untuk membaca jumlah kasus uji dari input.
15. `while (g--) { ... }`: Looping while untuk menjalankan setiap kasus uji.
16. `scanf("%d%d%d", &n, &m, &maxSum);`: Menggunakan scanf untuk membaca nilai n, m, dan maxSum dari input.
17. Mendeklarasikan array a dan b dengan ukuran sesuai dengan nilai n dan m yang telah dibaca sebelumnya.
18. Menggunakan loop for untuk membaca nilai elemen-elemen dari stack pertama (a) dan stack kedua (b) dari input.
19. `printf("%d\n", twoStacks(maxSum, a, n, b, m));`: Memanggil fungsi twoStacks untuk menghitung jumlah maksimum elemen yang dapat diambil dari kedua stack, dan mencetak hasilnya.
20. `return 0;`: Mengembalikan nilai 0, menandakan bahwa program telah berakhir dengan sukses.