

Nama:M.AGUNG BALYA

NIM:1203230095

Kelas:IF 03-01

No.1

Code

```
C Tugas_Queue_Praktikum.c C OTH1.c x C OTH2.c Tugas Queue.txt n.c
C OTH1.c > ...
1 #include <stdio.h>
2
3 struct Stone {
4     char* alphabet;
5     struct Stone* link;
6 };
7
8 void initialize(struct Stone* stone, char* letter) {
9     stone->link = NULL;
10    stone->alphabet = letter;
11 }
12
13 int main() {
14     struct Stone stone1, stone2, stone3, stone4, stone5, stone6, stone7, stone8, stone9;
15
16     initialize(&stone1, "F");
17     initialize(&stone2, "M");
18     initialize(&stone3, "A");
19     initialize(&stone4, "I");
20     initialize(&stone5, "K");
21     initialize(&stone6, "P");
22     initialize(&stone7, "U");
23     initialize(&stone8, "O");
24     initialize(&stone9, "R");
25
26     stone3.link = &stone6;
27     stone6.link = &stone9;
28     stone9.link = &stone4;
29     stone4.link = &stone7;
30     stone7.link = &stone1;
31     stone1.link = &stone8;
32     stone8.link = &stone2;
33     stone2.link = &stone5;
34     stone5.link = &stone3;
35
36     printf("%s", stone3.link->link->link->alphabet);
37     printf("%s", stone3.link->link->link->link->alphabet);
38 }
39
C Tugas_Queue_Praktikum.c C OTH1.c x C OTH2.c Tugas Queue.txt n.c
C OTH1.c > ...
13 int main() {
14     initialize(&stone1, "F");
15     initialize(&stone2, "M");
16     initialize(&stone3, "A");
17     initialize(&stone4, "I");
18     initialize(&stone5, "K");
19     initialize(&stone6, "P");
20     initialize(&stone7, "U");
21     initialize(&stone8, "O");
22     initialize(&stone9, "R");
23
24     stone3.link = &stone6;
25     stone6.link = &stone9;
26     stone9.link = &stone4;
27     stone4.link = &stone7;
28     stone7.link = &stone1;
29     stone1.link = &stone8;
30     stone8.link = &stone2;
31     stone2.link = &stone5;
32     stone5.link = &stone3;
33
34     printf("%s", stone3.link->link->link->alphabet);
35     printf("%s", stone3.link->link->link->link->alphabet);
36     printf("%s", stone3.link->link->link->link->link->alphabet);
37     printf("%s", stone3.link->link->link->link->link->link->alphabet);
38     printf("%s", stone3.link->link->alphabet);
39     printf("%s", stone3.link->alphabet);
40     printf("%s", stone3.link->link->link->alphabet);
41     printf("%s", stone3.link->link->link->link->alphabet);
42     printf("%s", stone3.link->link->link->link->link->link->link->alphabet);
43     printf("%s", stone3.alphabet);
44     printf("%s", stone3.link->link->link->alphabet);
45     printf("%s", stone3.link->link->link->link->link->link->link->link->alphabet);
46     printf("%s", stone3.alphabet);
47
48     printf("\n\nStack Visualization:\n\n");
49
50     struct Stone* current = &stone3;
51
52     return 0;
53 }
54
```

Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\agung\Februari\Week 7> cd "c:\agung\Februari\Week 7\" ; if ($?) { gcc OTH1.c -o OTH1 } ; if ($?) { .\OTH1 }
INFORMATIKA

Stack Visualization:
PS C:\agung\Februari\Week 7>
```

## Penjelasan

Mendeklarasikan sembilan variabel bertipe struct Stone yang mewakili sembilan batu.

Mencetak huruf-huruf yang terhubung dengan menggunakan pointer dan akses ke variabel alphabet dari batu-batu tersebut.

Menampilkan visualisasi stack dengan mendeklarasikan pointer ke struct Stone dan menginisiasinya dengan alamat dari batu ketiga.

No.2

## Kode

```
C Tugas_Queue_Praktikum.c C OTH1.c C OTH2.c X OTH2.txt C tempCodeRunner
C OTH2.c > main()
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX_SIZE 50
5
6 typedef struct {
7     int data[MAX_SIZE];
8     int top;
9 } Stack;
10
11 void initializeStack(Stack *stack) {
12     stack->top = -1;
13 }
14
15 void push(Stack *stack, int value) {
16     stack->data[++stack->top] = value;
17 }
18
19 int pop(Stack *stack) {
20     return stack->data[stack->top--];
21 }
22
23 int peek(Stack *stack) {
24     return stack->data[stack->top];
25 }
26
27 int isEmpty(Stack *stack) {
28     return stack->top == -1;
29 }
30
31 int processTwoStacks(int maxSum, int stackA[], int stackB[], int n, int m) {
32     Stack sA, sB;
33     initializeStack(&sA);
34     initializeStack(&sB);
35     for (int i = n - 1; i >= 0; i--) {
36         push(&sA, stackA[i]);
37     }
38
39     int total = 0, count = 0;
40     while (!isEmpty(&sA) || !isEmpty(&sB)) {
41         if (!isEmpty(&sA) && total + peek(&sA) <= maxSum) {
42             total += pop(&sA);
43             count++;
44         }
45         else if (!isEmpty(&sB) && total + peek(&sB) <= maxSum) {
46             total += pop(&sB);
47             count++;
48         }
49         else {
50             break;
51         }
52     }
53     return count;
54 }
55
56 int main() {
57     int numGames;
58     printf("Enter the number of games: ");
59     scanf("%d", &numGames);
60
61     while (numGames-- > 0) {
62         int n, m, maxSum;
63         printf("Enter the sizes of stacks A and B, and the maximum sum: ");
64         scanf("%d %d %d", &n, &m, &maxSum);
65
66         int stackA[n], stackB[m];
67         printf("Enter the elements of stack A: ");
68         for (int i = 0; i < n; i++) {
69             scanf("%d", &stackA[i]);
70         }
71
72         printf("Enter the elements of stack B: ");
73         for (int i = 0; i < m; i++) {
74             scanf("%d", &stackB[i]);
75         }
76
77         int result = processTwoStacks(maxSum, stackA, stackB, n, m);
78         printf("Result: %d\n", result);
79     }
80
81     return 0;
82 }
83
84 }
85
```

## Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS c:\agung\Februari\Week 7> cd "c:\agung\Februari\Week 7\" ; if ($?) { gcc 0TH2.c -o 0TH2 } ; if ($?) { .\0TH2 }
Enter the number of games: 5
Enter the sizes of stacks A and B, and the maximum sum: 4
4
3
Enter the elements of stack A: 4
5
1
1
Enter the elements of stack B: 3
1
1
2
Result: 1
```

## Penjelasan

`typedef struct {}` Ini mendefinisikan sebuah struktur data baru bernama `Stack`, yang terdiri dari array data dengan ukuran `MAX_SIZE` untuk menyimpan elemen-elemen tumpukan, dan variabel `top` yang menunjukkan indeks atas (terbaru) dari tumpukan.

`void initializeStack` Ini adalah fungsi yang digunakan untuk menginisialisasi tumpukan. Ketika dipanggil, variabel `top` dalam struktur `Stack` diatur menjadi `-1`, menunjukkan bahwa tumpukan kosong.

`void push` Ini adalah fungsi untuk menambahkan elemen baru ke dalam tumpukan. Nilai `value` ditambahkan ke dalam array data pada indeks `top` yang kemudian diinkremen, menunjukkan penambahan elemen baru ke dalam tumpukan.

`int peek` Ini adalah fungsi yang digunakan untuk melihat nilai teratas dari tumpukan tanpa menghapusnya. Ini hanya mengembalikan nilai dari array data pada indeks `top`.

`int isEmpty` Ini adalah fungsi yang mengembalikan nilai `1` jika tumpukan kosong (`top` sama dengan `-1`), dan `0` jika tidak.

Di dalam fungsi `main()`, program meminta input jumlah permainan (`numGames`), kemudian loop `while` akan berjalan sebanyak jumlah permainan tersebut.

Di dalam loop, program meminta input ukuran tumpukan dan nilai-nilai tumpukan A dan B untuk setiap permainan.

Setelah mendapatkan input, program memanggil fungsi `processTwoStacks()` untuk memproses kedua tumpukan, dan mencetak jumlah elemen yang diambil dari kedua tumpukan.