

## PENJELASAN SOAL LATIHAN ASD STRUCK AND STACK

### SOAL NOMER1

```
1  #include <stdio.h>
2
3  typedef struct Node {
4      struct Node *link;
5      char *alphabet;
6  } Node;
7
8  int main() {
9      Node 11, 12, 13, 14, 15, 16, 17, 18, 19;
10
11     11.link = NULL;
12     11.alphabet = "F";
13
14     12.link = NULL;
15     12.alphabet = "M";
16
17     13.link = NULL;
18     13.alphabet = "A";
19
20     14.link = NULL;
21     14.alphabet = "I";
22
23     15.link = NULL;
24     15.alphabet = "K";
25
26     16.link = NULL;
27     16.alphabet = "T";
28
29     17.link = NULL;
30     17.alphabet = "N";
31
32     18.link = NULL;
33     18.alphabet = "O";
34
35     19.link = NULL;
36     19.alphabet = "R";
37
38     17.link = &11;
39     11.link = &18;
40     18.link = &12;
41     12.link = &15;
42     15.link = &13;
43     13.link = &16;
44     16.link = &19;
45     19.link = &14;
46     14.link = &17;
47
48
49     printf("%s", 13.link->link->link->alphabet);
50     printf("%s", 13.link->link->link->link->alphabet);
51     printf("%s", 13.link->link->link->link->link->alphabet);
52     printf("%s", 13.link->link->link->link->link->link->alphabet);
53     printf("%s", 13.link->link->alphabet);
54     printf("%s", 13.link->link->link->link->link->link->link->alphabet);
55     printf("%s", 13.alphabet);
56     printf("%s", 13.link->alphabet);
57     printf("%s", 13.link->link->link->link->alphabet);
58     printf("%s", 13.link->link->link->link->link->link->link->link->alphabet);
59     printf("%s", 13.alphabet);
60
61     return 0;
62 }
63
```

1. Deklarasi Struktur Node: Struktur data Node dideklarasikan dengan dua anggota: link yang merupakan pointer ke node berikutnya dalam linked list, dan alphabet yang merupakan pointer ke string satu karakter.
2. Deklarasi Variabel Node: Dilakukan deklarasi sembilan variabel Node yang berturut-turut disebut l1 hingga l9, merepresentasikan simpul-simpul dalam linked list.
3. Inisialisasi Nilai: Nilai-nilai link dan alphabet dari setiap variabel Node diinisialisasi. Setiap alphabet diinisialisasi dengan string yang berisi satu huruf, sedangkan link diinisialisasi dengan NULL karena pada awalnya linked list masih kosong.
4. Penyusunan Linked List: Dilakukan penyusunan linked list dengan mengatur nilai link pada setiap node untuk menunjuk ke node berikutnya. Misalnya, l7.link diatur untuk menunjuk ke l1, l1.link diatur untuk menunjuk ke l8, dan seterusnya hingga semua node terhubung.
5. Cetak Huruf: Program mencetak beberapa huruf dari linked list dengan menggunakan pointer. Beberapa contoh cetakan huruf adalah:
  - `printf("%s", l3.link->link->link->alphabet);`: Mencetak huruf dari node ke-3 dari node yang ditunjuk oleh node l3.
  - `printf("%s", l3.link->link->link->link->alphabet);`: Mencetak huruf dari node ke-4 dari node yang ditunjuk oleh node l3.
  - Dan seterusnya hingga mencetak beberapa huruf sesuai dengan pengaturan pointer.

### HASIL OUTPUT

```
PS C:\Users\wicipto> cd "c:\Users\wicipto\Documents\ALGORITMA PEMROGRAMAN\SEMESTER 2\" ; if ($?) {
INFORMATIKA
PS C:\Users\wicipto\Documents\ALGORITMA PEMROGRAMAN\SEMESTER 2>
```

## SOAL NOMER 2

```
1  #include <stdio.h>
2
3  int twoStacks(int maxSum, int a[], int n, int b[], int m) {
4      int maxScore = 0, score = 0;
5      int i = 0, j = 0;
6      // Hitung berapa banyak elemen dari stack a yang dapat diambil tanpa melampaui maxSum
7      while (i < n && score + a[i] <= maxSum) {
8          score += a[i];
9          i++;
10     }
11     maxScore = i;
12     // Iterasi melalui stack b dan tambahkan elemen ke skor sambil memeriksa batasan maxSum
13     while (j < m && i >= 0) {
14         score += b[j];
15         j++;
16         // Kurangi elemen dari stack a jika diperlukan agar tidak melampaui maxSum
17         while (score > maxSum && i > 0) {
18             i--;
19             score -= a[i];
20         }
21         // Perbarui maxScore jika diperlukan
22         if (score <= maxSum && i + j > maxScore) {
23             maxScore = i + j;
24         }
25     }
26     return maxScore;
27 }
28
29 int main() {
30     int g;
31     scanf("%d", &g);
32     while (g--) {
33         int n, m, maxSum;
34         scanf("%d %d %d", &n, &m, &maxSum);
35         int a[n], b[m];
36         for (int i = 0; i < n; i++) {
37             scanf("%d", &a[i]);
38         }
39         for (int i = 0; i < m; i++) {
40             scanf("%d", &b[i]);
41         }
42         int result = twoStacks(maxSum, a, n, b, m);
43         printf("%d\n", result);
44     }
45     return 0;
46 }
47
```

### Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[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
```

1. Deklarasi Header: Diawali dengan inklusi header `stdio.h` yang digunakan untuk operasi masukan dan keluaran standar.
2. Fungsi `twoStacks`: Fungsi ini mengambil input berupa nilai maksimum (`maxSum`), dua array integer (`a[]` dan `b[]`), serta panjang masing-masing array (`n` dan `m`). Fungsi ini bertujuan untuk menghitung maksimum jumlah elemen yang dapat diambil dari kedua stack sedemikian hingga total nilai mereka tidak melebihi `maxSum`.
  - `maxScore` dan `score` diinisialisasi dengan nilai 0, `i` dan `j` digunakan sebagai indeks untuk iterasi melalui stack `a` dan `b`.
  - Iterasi pertama dilakukan melalui stack `a`, menambahkan nilai-nilai elemen dari `a[]` ke `score` sampai batas `maxSum` tercapai atau tidak dapat menambah lagi.
  - `maxScore` diatur menjadi `i`, yang merupakan jumlah elemen yang telah diproses dari stack `a`.
  - Iterasi kedua dilakukan melalui stack `b`, menambahkan nilai-nilai elemen dari `b[]` ke `score` dan mengurangi elemen dari `a[]` jika diperlukan agar total tidak melebihi `maxSum`.
  - Pada setiap iterasi, nilai `maxScore` diperbarui jika jumlah elemen dari kedua stack saat itu lebih besar daripada nilai sebelumnya.
  - Fungsi mengembalikan nilai `maxScore` yang merupakan maksimum jumlah elemen yang dapat diambil dari kedua stack tanpa melebihi `maxSum`.
3. Fungsi `main`: Fungsi utama dari program.
  - Mendeklarasikan variabel `g` untuk menyimpan jumlah kasus yang akan diuji.
  - Menggunakan perulangan `while` untuk menjalankan proses uji sebanyak `g` kali.
  - Untuk setiap kasus uji, membaca masukan `n`, `m`, dan `maxSum`.
  - Membaca nilai-nilai elemen dari stack `a[]` dan `b[]`.
  - Memanggil fungsi `twoStacks` dengan parameter yang sesuai dan mencetak hasilnya.

### HASIL OUTPUT

```
PS C:\Users\wicipto> cd "c:\Users\wicipto\Documents\ALGORITMA PEMROGRAMAN\SEMESTER 2\" ;  
}  
1  
5 4 10  
4 2 4 6 1  
2 1 8 5  
4  
PS C:\Users\wicipto\Documents\ALGORITMA PEMROGRAMAN\SEMESTER 2> █
```