Struct dan Stack

Nama : I Gede Fender Adrea Veda
NIM : 1203230009
Kelas : IF 03-01

1. Source Code

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

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

int main() {

    struct Stone l1, l2, l3, l4, l5, l6, l7, l8, l9;

    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;
    l9.alphabet = "R";

l3.link = &l6; //A->T
l6.link = &l9; //T->R
```

```
42. l9.link = &l4; //R->I
43. l4.link = &l7; //I->N
44. l7.link = &l1; //N->F
45. l1.link = &l8; //T->O
46. l8.link = &l2; //O->M
47. l2.link = &l5; //M->K
48. l5.link = &l3; //K->A
49. // Akses data menggunakan l3 sebagai titik awal
50.
51. printf(" %s", l3.link->link->link->alphabet);
52. printf(" %s", l3.link->link->link->link->alphabet);
53. printf(" %s", l3.link->link->link->link->link->alphabet);
54. printf(" %s", l3.link->link->link->link->link->link->alphabet);
55. printf(" %s", l3.link->link->alphabet);
56. printf(" %s", l3.link->link->link->link->link->link->link->alphabet);
57. printf(" %s", l3.link->link->link->link->link->link->link->link->link-
    >alphabet);
58. printf(" %s", l3.link->alphabet);
59. printf(" %s", l3.link->link->link->alphabet);
60. printf(" %s", l3.link->link->link->link->link->link->link->link->alphabet);
61. printf(" %s", l3.link->link->link->link->link->link->link->link->link-
    >alphabet);
62.

63. struct Stone* current = &l3;
64.
65. return 0;
66. }
```

Output

Penjelasan

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

Merupakan header file yang biasa digunakan dalam Bahasa C

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

Terdapat struktur Stone yang memiliki anggota **char* alphabet;** pointer yang artinya menyimpan sebuah karakter atau string yang dimana dalam program ini abjad (alphabet). **struct Stone* link;** merupakan pointer ke struktur Stone. Anggota ini digunakan untuk

membuat hubungan antara dua elemen Stone dan membentuk linked list. Menunjuk ke elemen Stone berikutnya dalam linked list.

```
int main() {

    struct Stone l1, l2, l3, l4, l5, l6, l7, l8, l9;

    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;
    l9.alphabet = "R";
```

- Variabel bertipe struct Stone yang masing – masing merupakan elemen linked list yang di dalamnya terdapat **link** dan **alphabet.**
- **link** merupakan pointer ke elemen struct stone selanjutnya dalam linked list yang awalnya diarahkah ke **NULL** (akhir linked list).
- **alphabet** pointer ke string atau struktur untuk menyimpan abjad.
- Inisialisasi variable l1 sampai l9 merupakan isian elemen – elemen linked list yang terhubung dengan elemen selanjutnya dan membentuk linked list yang terdiri dari huruf – huruf tersebut.

```
l3.link = &l6; //A->T
l6.link = &l9; //T->R
l9.link = &l4; //R->I
l4.link = &l7; //I->N
l7.link = &l1; //N->F
l1.link = &l8; //T->O
l8.link = &l2; //O->M
```

```
l2.link = &l5; //M->K
l5.link = &l3; //K->A
```

Operasi yang mengatur pointer link dari satu elemen linked list ke elemen linked list lainnya. Sesuai urutan yang diberikan, mendefinisikan hubungan antara elemen-elemen linked list yang membentuk lingkaran huruf berawal dari A dan Kembali ke A.

```
printf(" %s", l3.link->link->link->alphabet);
printf(" %s", l3.link->link->link->link->alphabet);
printf(" %s", l3.link->link->link->link->link->alphabet);
printf(" %s", l3.link->link->link->link->link->link->alphabet);
printf(" %s", l3.link->link->alphabet);
printf(" %s", l3.link->link->link->link->link->link->link->alphabet);
printf(" %s", l3.link->link->link->link->link->link->link->link->link-
>alphabet);
printf(" %s", l3.link->alphabet);
printf(" %s", l3.link->link->link->alphabet);
printf(" %s", l3.link->link->link->link->link->link->link->link->link->alphabet);
printf(" %s", l3.link->link->link->link->link->link->link->link->link-
>alphabet);
```

printf digunakan untuk mencetak huruf-huruf yang terdapat dalam linked list berdasarkan hubungan yang sudah dibuat. Misalnya **printf %s", l3.link->link->link->alphabet)** : Mencetak huruf yang terdapat 3 langkah ke depan dari huruf 'A' (A->T->R->I). Huruf akan selalu menjadi yang pertama **l3.link->** diakses.

```
struct Stone* current = &l3;
```

current akan berisi alamat dari objek **Stone** yang terletak di variabel **l3**. Artinya **current** dan **l3** objek yang sama yaitu perubahan yang dilakukan pada **current** akan berpengaruh pada **l3** dan sebaliknya.

```
return 0;
```

Program selesai dijalankan.

2. Source Code

```
1.  #include <stdio.h>
2.
3.  int twoStacks(int maxSum, int a[], int n, int b[], int m) {
4.      int sum_a[n+1];
5.      int sum_b[m+1];
6.
7.      sum_a[0] = 0;
8.      sum_b[0] = 0;
9.
10.     for (int i = 1; i <= n; i++) {
11.         sum_a[i] = sum_a[i-1] + a[i-1];
12.     }
13.
14.     for (int i = 1; i <= m; i++) {
```

```
        sum_b[i] = sum_b[i-1] + b[i-1];
    }

    int ans = 0;
    int j = m;

    for (int i = 0; i <= n; i++) {
        if (sum_a[i] > maxSum) {
            break;
        }
        while (sum_a[i] + sum_b[j] > maxSum) {
            j--;
        }
        if (i + j > ans) {
            ans = i + j;
        }
    }

    return ans;
}

int main() {
    int t;
    scanf("%d", &t);

    for (int i = 0; i < t; i++) {
        int n, m, maxSum;
        scanf("%d %d %d", &n, &m, &maxSum);

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

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

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

    return 0;
}
```

Output

Visualisasi
Langkah 1: Stack A: Stack B:
Langkah 2 (Push A): Stack A: 4 Stack B:
Langkah 3 (Push A): Stack A: 4 5 Stack B:
Langkah 4 (Push A): Stack A: 4 5 2 Stack B:
Langkah 5 (Push A): Stack A: 4 5 2 1 Stack B
Langkah 6 (Push A): Stack A: 4 5 2 1 1 Stack B
Langkah 7 (Push B): Stack A: 4 5 2 1 1 Stack B: 3
Langkah 8 (Push B): Stack A: 4 5 2 1 1 Stack B: 3 1
Langkah 9 (Push B): Stack A: 4 5 2 1 1 Stack B: 3 1 1
Langkah 10 (Push B): Stack A: 4 5 2 1 1 Stack B: 3 1 1 2
Langkah 11 (Pop dari B): Stack A: 4 5 2 1 1 Stack B: 3 1 1
Langkah 12 (Pop dari B): Stack A: 4 5 2 1 1 Stack B: 3 1
Langkah 13 (Pop dari A): Stack A: 4 5 2 1 Stack B: 3 1
Langkah 14 (Pop dari B): Stack A: 4 5 2 1 Stack B: 3
Langkah 15 (Pop dari B): Stack A: 4 5 2 1 Stack B:
Langkah 16 (Pop dari B): Stack A: 4 5 2 Stack B:
Langkah 17 (Pop dari A): Stack A: 4 5 Stack B:
Langkah 18 (Pop dari B): Stack A: 4 5 Stack B:
Langkah 19 (Pop dari A): Stack A: 4 Stack B:
Langkah 20 (Pop dari A): Stack A: Stack B:

Penjelasan

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

Merupakan header file yang biasa digunakan dalam Bahasa C

```c
int twoStacks(int maxSum, int a[], int n, int b[], int m) {
    int sum_a[n+1];
    int sum_b[m+1];
```

Ada dua array bernama **sum_a** dan **sum_b**. Array **sum_a** digunakan untuk menyimpan penjumlahan setiap elemen dalam array **a**, begitu pun dengan array **b**. **n** dan **m** merupakan panjang array dengan panjang maksimumnya **n+1** dan **m+1**.

```c
sum_a[0] = 0;
sum_b[0] = 0;
```

Memulai nilai awal masing – masing array.

```c
for (int i = 1; i <= n; i++) {
        sum_a[i] = sum_a[i-1] + a[i-1];
    }
```

```
    for (int i = 1; i <= m; i++) {
        sum_b[i] = sum_b[i-1] + b[i-1];
    }
```

Perulangan pertama variabel **i** dari 1 sampai n+1 akan membuat array **sum_a** dengan cara memanggil fungsi sum_**a[i]** sebagai hasil dari **sum_a[i-1]** yang merupakan hasil dari perulangan sebelumnya, ditambah dengan **a[i-1]**. Artinya setiap **sum_a[i]** merupakan jumlah dari elemen dari array **a** dari indeks **0** sampai dengan indeks **i-1**. Begitupun dengan **m** untuk array **b**. Intinya **sum_a** dan **b** akan berisi penjumlahan dari masing – masing array nya.

```
int ans = 0;
int j = m;
```

Variable **ans** digunakan untk menyimpan hasil perhitungan, diinisialisasi **0** karena akan diawali **0**. Sedangkan variable **j** digunakan utuk indeks array **b**. Diinisialisasi **m** karena merupakan panjang dari array **b**.

```
for (int i = 0; i <= n; i++) {
        if (sum_a[i] > maxSum) {
            break;
        }
        while (sum_a[i] + sum_b[j] > maxSum) {
            j--;
        }
        if (i + j > ans) {
            ans = i + j;
        }
    }

    return ans;
```

- Perulangan diatas berfungsi untuk mencari urutan pilihan yang dapat membentuk jumlah terbesar yang lebih kecil dari maxSum. Variabel **i** dari **0** sampai n+1 akan memeriksa setiap elemen dari array **a** apakah **sum_a[i]** lebih besar dari **maxSum**. Jika iya maka akan dihentikan oleh **break**.
- Selanjutnya memeriksa apakah **sum_a[i]** + **sum_b[j]** lebih besar dari **maxSum**. Jika iya maka akan memeriksa elemen sebelumnya array **b** dengan mengurangi **j**.
- Jika **i** + **j** lebih besar dari **ans**, maka akan diubah dengan **i** + **j**. Artinya **ans** akan diisi urutan yang dapat membentuk jumlah terbesar namun lebih kecil dari **maxSum**.
- Setelah perulangan selesai nilai **ans** dikembalikan.

```
int main() {
    int t;
    scanf("%d", &t);
```

- Variabel **t** akan menyimpan jumlah soal yang akan dijawab.
- **%d** artinya user harus memasukkan bilangan bulat untuk soal.

- **&t** alamat variable **t**.

```
for (int i = 0; i < t; i++) {
        int n, m, maxSum;
        scanf("%d %d %d", &n, &m, &maxSum);
```

- Perulangan **i** dari **t** sampai t-1 ada membaca input soal
- Membaca nilai variable **n**, **m**, dan **maxSum**.
- **scanf** meminta user memasukkan tiga nilai untuk variabel **n**, **m**, dan **maxSum**, dan nilai yang diinputkan akan disimpan ke dalam variabel **n**, **m**, dan **maxSum**.
- "%d %d %d" menunjukkan bahwa user harus memasukkan tiga nilai bulat terpisah dengan spasi. Artinya nilai **n**, **m** dan **maxSum** dipisahkan dengan spasi.

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

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

- Perulangan **for** dengan variabel **j** dari 0 sampai n-1 akan membaca input untuk setiap elemen dari array **a**.
- **scanf** meminta user memasukkan nilai untuk elemen dari array **a**, dan nilai yang diinputkan disimpan ke dalam elemen dari array **a**.
- string **%d** menunjukkan bahwa user harus memasukkan sebuah bilangan bulat dan dipisahkan dengan spasi.
- Begitupun untuk array **b** dengan variable **m**.

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

- Fungsi **twoStacks** dijalankan dengan parameter **maxSum**, **a**, **n**, **b**, dan **m**, dan nilai yang disimpan ke dalam variabel **result** akan dikembalikan.
- **printf** dengan format string **%d\n** akan menampilkan nilai dari variabel **result**.

```
return 0;
```

Program selesai dijalankan