

**LAPORAN PRAKTIKUM**

**OTH STRUCT & STACK**



Dibuat oleh

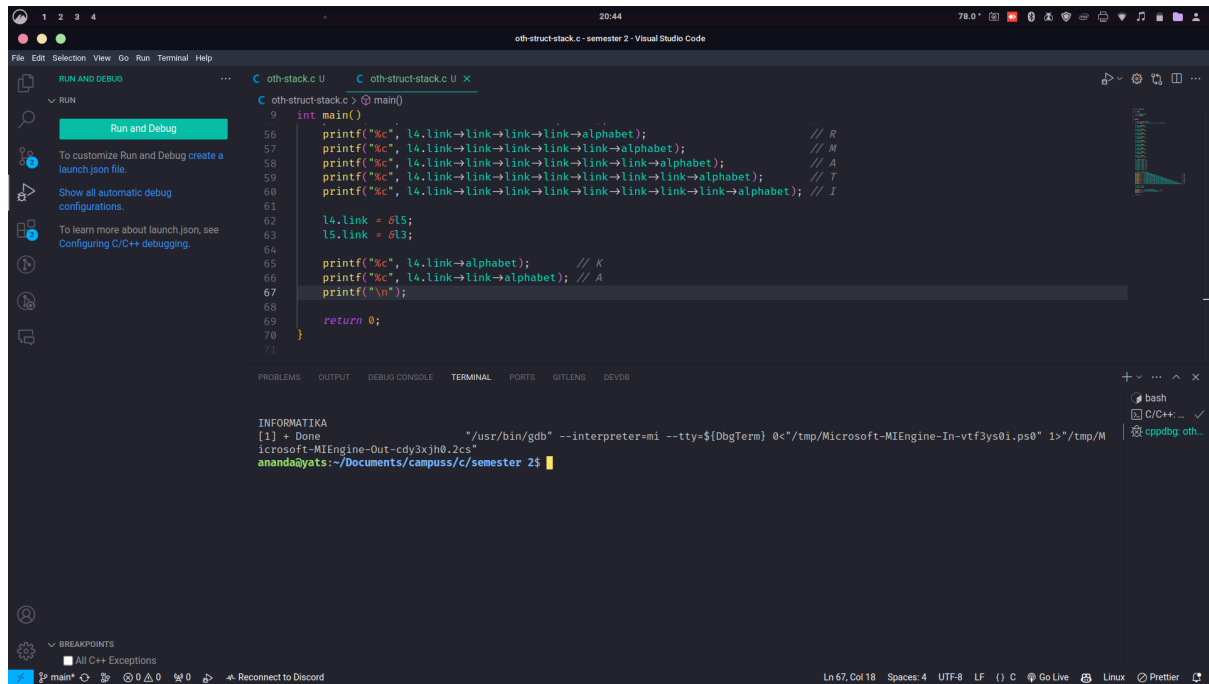
Ananda Bintang Saputra ( 1203230040 )

**Fakultas Informatika**

**Prodi Informatika**

**Telkom University Surabaya 2023/2024**

## Output Program



Game of Two Stacks | Hackerrank

```
235 int parse_int(char *str)
236 {
237     char *endptr;
238     int value = strtol(str, &endptr, 10);
239
240     if (endptr == str || *endptr != '\0')
241     {
242         exit(EXIT_FAILURE);
243     }
244     return value;
245 }
247
```

Line: 247 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

### Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)

```
1 1
2 5410
3 42461
4 2185
```

Your Output (stdout)

```
1 4
```

Expected Output

```
1 4
```

```
237 char *endptr;
238 int value = strtol(str, &endptr, 10);
239
240 if (endptr == str || *endptr != '\0')
241 {
242     exit(EXIT_FAILURE);
243 }
244 return value;
245 }
247
```

Line: 247 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Test case 0

Compiler Message

Success

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Input (stdin)

```
1 1
2 5410
3 42461
4 2185
```

Expected Output

```
1 4
```

## Penjelasan Kode

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

Membuat tipe data baru berbentuk struct yang memiliki elemen "char alphabet" dan "struct node \*link", elemen struct berfungsi untuk menunjuk address dari node link dirinya sendiri (nested)

```

// Node initialization
struct node 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';

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

```

Menginisialisasi variabel bertipe data “struct node” l1-l9 untuk assign setiap huruf. Lalu link/menghubungkan setiap nodes agar menjadi kata yang diinginkan.

```

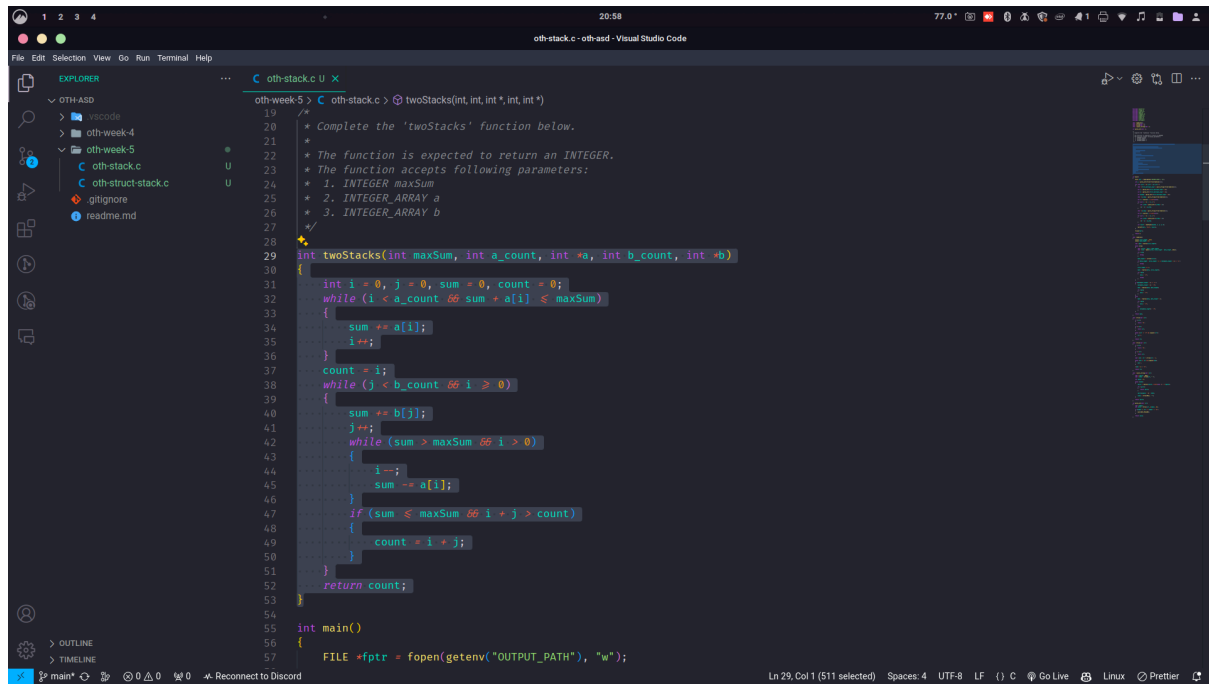
// Print linked list
printf("%c", l4.alphabet); // I
printf("%c", l4.link→alphabet); // N
printf("%c", l4.link→link→alphabet); // F
printf("%c", l4.link→link→link→alphabet); // O
printf("%c", l4.link→link→link→link→alphabet); // R
printf("%c", l4.link→link→link→link→link→alphabet); // M
printf("%c", l4.link→link→link→link→link→link→alphabet); // A
printf("%c", l4.link→link→link→link→link→link→link→alphabet); // T
printf("%c", l4.link→link→link→link→link→link→link→link→alphabet); // I

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

printf("%c", l4.link→alphabet); // K
printf("%c", l4.link→link→alphabet); // A
printf("\n");

```

Output nodes yang telah dilink/dihubungkan menjadi satu kesatuan kata.



```
19 /*
20  * Complete the 'twoStacks' function below.
21  *
22  * The function is expected to return an INTEGER.
23  * The function accepts following parameters:
24  * 1. INTEGER maxSum
25  * 2. INTEGER_ARRAY a
26  * 3. INTEGER_ARRAY b
27  */
28
29 int twoStacks(int maxSum, int a_count, int *a, int b_count, int *b)
30 {
31     int i = 0, j = 0, sum = 0, count = 0;
32     while (i < a_count && sum + a[i] <= maxSum)
33     {
34         sum += a[i];
35         i++;
36     }
37     count = i;
38     while (j < b_count && i > 0)
39     {
40         sum += b[j];
41         j++;
42         while (sum > maxSum && i > 0)
43         {
44             i--;
45             sum -= a[i];
46         }
47         if (sum <= maxSum && i + j > count)
48         {
49             count = i + j;
50         }
51     }
52     return count;
53 }
54
55 int main()
56 {
57     FILE *fptr = fopen(getenv("OUTPUT_PATH"), "w");
```

### 1. Iterasi melalui stack pertama:

Kode ini mengiterasi elemen-elemen stack pertama (a) selama jumlah elemen (sum) tidak melebihi maxSum.

Di setiap iterasi, kode menambahkan elemen saat ini dari stack pertama ke sum dan meningkatkan penghitung i.

Loop ini melacak berapa banyak elemen yang dapat diambil dari stack pertama tanpa melebihi maxSum.

### 2. Inisialisasi count:

Variabel count diinisialisasi dengan nilai i saat ini. Ini mewakili jumlah maksimum elemen yang dapat diambil dari stack pertama sejauh ini.

### 3. Iterasi melalui stack kedua:

Kode kemudian mengiterasi elemen-elemen stack kedua (b).

Di setiap iterasi, kode menambahkan elemen saat ini dari stack kedua ke sum dan meningkatkan penghitung j.

#### 4. Menyesuaikan elemen dari stack pertama (jika perlu):

Di dalam loop kedua, ada loop bersarang yang beriterasi selama sum melebihi maxSum dan ada elemen di stack pertama (i lebih besar dari 0).

Di setiap iterasi loop bersarang, kode ini menghapus elemen terakhir dari stack pertama dan mengurangi nilainya dari sum. Ini memastikan bahwa sum tidak melebihi maxSum.

#### 5. Perbarui count (opsional):

Setelah menyesuaikan elemen dari stack pertama (jika perlu), kode ini memeriksa apakah sum saat ini kurang dari atau sama dengan maxSum dan jumlah gabungan elemen dari kedua stack ( $i + j$ ) lebih besar dari count saat ini.

Jika kedua kondisi benar, kode ini memperbarui count ke jumlah elemen gabungan ( $i + j$ ). Ini memastikan bahwa count mewakili jumlah maksimum elemen yang dapat diambil dari kedua stack tanpa melebihi maxSum.

#### 6. Return count:

Setelah mengiterasi kedua stack, fungsi ini mengembalikan nilai akhir count, yang mewakili jumlah maksimum elemen yang dapat diambil dari kedua stack tanpa melebihi maxSum.