

Nama : M Alief Alfaridzi

Nim : 1203230045

Kelas : IF 03-03

Tugas OTH Struct dan Stack

Soal No 1

Source Code

```
1  #include <stdio.h>
2
3  struct link_t {
4      char alphabet;
5      struct link_t *link;
6  };
7
8  int main() {
9      // initialize the nodes
10     struct link_t l1, l2, l3, l4, l5, l6, l7, l8, l9;
11
12     l1.link = NULL;
13     l1.alphabet = 'F';
14
15     l2.link = NULL;
16     l2.alphabet = 'M';
17
18     l3.link = NULL;
19     l3.alphabet = 'A';
20
21     l4.link = NULL;
22     l4.alphabet = 'I';
23
24     l5.link = NULL;
25     l5.alphabet = 'K';
26
27     l6.link = NULL;
28     l6.alphabet = 'T';
29
30     l7.link = NULL;
31     l7.alphabet = 'N';
32
33     l8.link = NULL;
34     l8.alphabet = 'O';
35
36     l9.link = NULL;
37     l9.alphabet = 'R';
38
39     // linking the nodes
40     l3.link = &l6;
41     l6.link = &l9;
42     l9.link = &l4;
43     l4.link = &l7;
44     l7.link = &l1;
45     l1.link = &l8;
46     l8.link = &l2;
47     l2.link = &l5;
48     l5.link = &l3;
49
50     // starting point from l3
51     printf("Output: %c%c%c%c%c%c%c%c%c\n",
52         l3.link->link->link->alphabet, // I
53         l3.link->link->link->link->alphabet, // N
54         l3.link->link->link->link->link->alphabet, // F
55         l3.link->link->link->link->link->link->alphabet, // O
56         l3.link->link->alphabet, // R
57         l3.link->link->link->link->link->link->link->alphabet, // M
58         l3.alphabet, // A
59         l3.link->alphabet, // T
60         l3.link->link->link->alphabet, // I
61         l3.link->link->link->link->link->link->link->link->alphabet, // K
62         l3.alphabet // A
63     );
64
65     return 0;
66 }
```

Penjelasan Code

Berikut ini penjelasan code program persatu baris atau perbaris:

- Baris (1) `#include <stdio.h>` Mengimpor header file `stdio.h` yang berisi deklarasi fungsi standar untuk input dan output seperti `printf` dan `scanf`.
- Baris (3) `struct link_t {` Baris ini merupakan awal dari definisi sebuah struktur dengan nama `link_t`. Kata kunci `struct` digunakan untuk mendefinisikan sebuah struktur. `link_t` adalah nama dari struktur yang sedang didefinisikan. `{` menandakan awal dari definisi struktur.
- Baris (4) `char alphabet;` Ini adalah deklarasi anggota pertama dari struktur `link_t`. Anggota ini memiliki tipe data `char` yang merupakan tipe data untuk menyimpan satu karakter. Dalam hal ini, anggota ini disebut `alphabet` dan akan menyimpan sebuah karakter.
- Baris (5) `struct link_t *link;` Ini adalah deklarasi anggota kedua dari struktur `link_t`. Anggota ini adalah pointer ke struktur `link_t` itu sendiri, sehingga kita bisa membuat linked list dengan menggunakan struktur ini. `struct link_t *` menunjukkan bahwa `link` adalah pointer yang menunjuk ke suatu objek dengan tipe data `struct link_t`.
- Baris (6) `};` Baris ini menandakan akhir dari definisi struktur `link_t`. Kurung kurawal `}` menutup definisi struktur tersebut.
- Baris (8) `int main()` Menandakan awal dari fungsi utama, yang merupakan titik masuk program.
- Baris (10) `struct link_t l1, l2, l3, l4, l5, l6, l7, l8, l9;` Baris ini mendeklarasikan sembilan variabel bertipe `struct link_t`, yaitu `l1, l2, l3, l4, l5, l6, l7, l8, l9`. Variabel-variabel ini digunakan untuk merepresentasikan node-node dalam struktur data yang mungkin merupakan bagian dari linked list.
- Baris (12) `l1.link = NULL;` Ini mengatur anggota `link` dari node `l1` menjadi `NULL`. Ini menunjukkan bahwa `l1` adalah node pertama dalam linked list dan tidak menunjuk ke node mana pun.
- Baris (13) `l1.alphabet = 'F';` Ini mengatur anggota `alphabet` dari node `l1` menjadi karakter `'F'`.
- Baris (15) `l2.link = NULL;` Ini mengatur anggota `link` dari node `l2` menjadi `NULL`.
- Baris (16) `l2.alphabet = 'M';` Ini mengatur anggota `alphabet` dari node `l2` menjadi karakter `'M'`.
- Baris (18) `l3.link = NULL;` Ini mengatur anggota `link` dari node `l3` menjadi `NULL`.

- Baris (19) 13.alphabet = 'A'; Ini mengatur anggota alphabet dari node 13 menjadi karakter 'A'.
- Baris (21) 14.link = NULL; Ini mengatur anggota link dari node 14 menjadi NULL.
- Baris (22) 14.alphabet = 'I'; Ini mengatur anggota alphabet dari node 14 menjadi karakter 'I'.
- Baris (24) 15.link = NULL; Ini mengatur anggota link dari node 15 menjadi NULL.
- Baris (25) 15.alphabet = 'K'; Ini mengatur anggota alphabet dari node 15 menjadi karakter 'K'.
- Baris (27) 16.link = NULL; Ini mengatur anggota link dari node 16 menjadi NULL.
- Baris (28) 16.alphabet = 'T'; Ini mengatur anggota alphabet dari node 16 menjadi karakter 'T'.
- Baris (30) 17.link = NULL; Ini mengatur anggota link dari node 17 menjadi NULL.
- Baris (31) 17.alphabet = 'N'; Ini mengatur anggota alphabet dari node 17 menjadi karakter 'N'.
- Baris (33) 18.link = NULL; Ini mengatur anggota link dari node 18 menjadi NULL.
- Baris (34) 18.alphabet = 'O'; Ini mengatur anggota alphabet dari node 18 menjadi karakter 'O'.
- Baris (36) 19.link = NULL; Ini mengatur anggota link dari node 19 menjadi NULL.
- Baris (37) 19.alphabet = 'R'; Ini mengatur anggota alphabet dari node 19 menjadi karakter 'R'.
- Baris (40) 13.link = &l6; Ini mengatur link dari node 13 untuk menunjuk ke alamat node 16, sehingga node 13 terhubung ke node 16.
- Baris (41) 16.link = &l9; Ini mengatur link dari node 16 untuk menunjuk ke alamat node 19, sehingga node 16 terhubung ke node 19.
- Baris (42) 19.link = &l4; Ini mengatur link dari node 19 untuk menunjuk ke alamat node 14, sehingga node 19 terhubung ke node 14.
- Baris (43) 14.link = &l7; Ini mengatur link dari node 14 untuk menunjuk ke alamat node 17, sehingga node 14 terhubung ke node 17.
- Baris (44) 17.link = &l1; Ini mengatur link dari node 17 untuk menunjuk ke alamat node 11, sehingga node 17 terhubung ke node 11.
- Baris (45) 11.link = &l8; Ini mengatur link dari node 11 untuk menunjuk ke alamat node 18, sehingga node 11 terhubung ke node 18.
- Baris (46) 18.link = &l2; Ini mengatur link dari node 18 untuk menunjuk ke alamat node 12, sehingga node 18 terhubung ke node 12.

- Baris (47) `l2.link = &l5`; Ini mengatur link dari node l2 untuk menunjuk ke alamat node l5, sehingga node l2 terhubung ke node l5.
- Baris (48) `l5.link = &l3`; Ini mengatur link dari node l5 untuk menunjuk ke alamat node l3, sehingga node l5 terhubung kembali ke node l3, melengkapi siklus dan membentuk linked list yang lengkap.
- Baris (51) `printf("Output: %c%c%c%c%c%c%c%c%c%c%c%c\n",` Baris ini merupakan awal dari fungsi `printf()` yang akan mencetak output ke layar. Format string `"Output: %c%c%c%c%c%c%c%c%c%c%c%c\n"` menentukan format output yang akan dicetak. `%c` adalah placeholder untuk karakter.
- Baris (52) `l3.link->link->link->alphabet`, Baris ini mengakses nilai dari anggota `alphabet` dari node yang terhubung secara berurutan dari l3. `l3.link` mengarahkan ke node selanjutnya dari l3, Karakter pertama yang dicetak adalah `l3.link->link->link->alphabet`, yang sesuai dengan huruf 'T'.
- Baris (53) `l3.link->link->link->link->alphabet`, Baris ini mengakses nilai dari anggota `alphabet` dari node yang terhubung secara berurutan dari l3. `l3.link` mengarahkan ke node selanjutnya dari l3, Karakter kedua adalah `l3.link->link->link->link->alphabet`, yang sesuai dengan huruf 'N'.
- Baris (54) `l3.link->link->link->link->link->alphabet`, Baris ini mengakses nilai dari anggota `alphabet` dari node yang terhubung secara berurutan dari l3. `l3.link` mengarahkan ke node selanjutnya dari l3, Karakter ketiga adalah `l3.link->link->link->link->link->alphabet`, yang sesuai dengan huruf 'F'.
- Baris (55) `l3.link->link->link->link->link->link->alphabet`, Baris ini mengakses nilai dari anggota `alphabet` dari node yang terhubung secara berurutan dari l3. `l3.link` mengarahkan ke node selanjutnya dari l3, Karakter keempat adalah `l3.link->link->link->link->link->link->alphabet`, yang sesuai dengan huruf 'O'.
- Baris (56) `l3.link->link->alphabet`, Baris ini mengakses nilai dari anggota `alphabet` dari node yang terhubung secara berurutan dari l3. `l3.link` mengarahkan ke node selanjutnya dari l3, Karakter kelima adalah `l3.link->link->alphabet`, yang sesuai dengan huruf 'R'.
- Baris (57) `l3.link->link->link->link->link->link->link->alphabet`, Baris ini mengakses nilai dari anggota `alphabet` dari node yang terhubung secara berurutan dari l3. `l3.link` mengarahkan ke node selanjutnya dari l3, Karakter keenam adalah `l3.link->link->link->link->link->link->link->alphabet`, yang sesuai dengan huruf 'M'.

- Baris (58) l3.alphabet, Baris ini mengakses nilai dari anggota alphabet dari node yang terhubung secara berurutan dari l3. Karakter ketujuh adalah l3.alphabet, yang sesuai dengan huruf 'A'.
- Baris (59) l3.link->alphabet, Baris ini mengakses nilai dari anggota alphabet dari node yang terhubung secara berurutan dari l3. l3.link mengarahkan ke node selanjutnya dari l3, Karakter kedelapan adalah l3.link->alphabet, yang sesuai dengan huruf 'T'.
- Baris (60) l3.link->link->link->alphabet, Baris ini mengakses nilai dari anggota alphabet dari node yang terhubung secara berurutan dari l3. l3.link mengarahkan ke node selanjutnya dari l3, Karakter kesembilan adalah l3.link->link->link->alphabet, yang sesuai dengan huruf 'I'.
- Baris (61) l3.link->link->link->link->link->link->link->alphabet, Baris ini mengakses nilai dari anggota alphabet dari node yang terhubung secara berurutan dari l3. l3.link mengarahkan ke node selanjutnya dari l3, Karakter kesepuluh adalah l3.link->link->link->link->link->link->link->alphabet, yang sesuai dengan huruf 'K'.
- Baris (62) l3.alphabet, Baris ini mengakses nilai dari anggota alphabet dari node yang terhubung secara berurutan dari l3. Karakter kesebelas adalah l3.alphabet, yang sesuai dengan huruf 'A'.
- Baris (63)); Mengakhiri cetak output link.
- Baris (65) return 0; Mengakhiri fungsi utama dan mengembalikan nilai 0, yang menunjukkan keberhasilan penyelesaian program.
- Baris (66) } Mengakhiri fungsi main utama program.

SS Input

```
l3.link->link->link->alphabet, // I
l3.link->link->link->link->alphabet, // N
l3.link->link->link->link->link->alphabet, // F
l3.link->link->link->link->link->link->alphabet, // O
l3.link->link->alphabet, // R
l3.link->link->link->link->link->link->link->alphabet, // M
l3.alphabet, // A
l3.link->alphabet, // T
l3.link->link->link->alphabet, // I
l3.link->link->link->link->link->link->link->link->alphabet, // K
l3.alphabet // A
```

SS Output

```
PS C:\Users\HP\OneDrive\Documents\ALGOSDAT> cd "c:\Users\HP\OneDrive\Documents\ALGOSDAT\" ; if ($?) { gcc Soal10TH1StructdanStack.c -o Soal10TH1StructdanStack } ; if ($?) { .\Soal10TH1StructdanStack }
Output: INFORMATIKA
PS C:\Users\HP\OneDrive\Documents\ALGOSDAT>
```

Soal No 2

Hackerrank : Sukses

Congratulations

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

[Next Challenge](#)

Game of Two Stacks ★

[Problem](#) | [Submissions](#) | [Leaderboard](#) | [Discussions](#) | [Editorial](#)

You made this submission 21 hours ago.

Score: 30.00 **Status:** Accepted

People who solved **Game of Two Stacks** attempted this next:

Penjelasan Code Program

Berikut ini penjelasan code program persatu baris atau perbaris:

- Baris (1) `#include <stdio.h>` Mengimpor header file `stdio.h` yang berisi deklarasi fungsi standar untuk input dan output seperti `printf` dan `scanf`.
- Baris (2) `#include <stdlib.h>` Baris ini berisi file header perpustakaan standar, yang menyediakan fungsi untuk operasi tujuan umum seperti alokasi memori, konversi tipe, dan pembuatan angka acak.
- Baris (3) `#include <string.h>` Baris ini menyertakan file header perpustakaan string, yang menyediakan fungsi untuk memanipulasi string.
- Baris (4) `#include <ctype.h>` Baris ini berisi file header perpustakaan tipe karakter, yang menyediakan fungsi untuk menguji dan memetakan karakter.
- Baris (6) `char* readline();` Baris ini mendeklarasikan prototipe fungsi untuk nama fungsi `readline` yang mengembalikan pointer ke karakter.
- Baris (7) `char* ltrim(char*);` Baris ini mendeklarasikan prototipe fungsi untuk nama fungsi `ltrim` yang mengambil pointer ke karakter sebagai argumen dan mengembalikan pointer ke karakter.
- Baris (8) `char* rtrim(char*);` Baris ini mendeklarasikan prototipe fungsi untuk nama fungsi `rtrim` yang mengambil pointer ke karakter sebagai argumen dan mengembalikan pointer ke karakter.

- Baris (9) `char** split_string(char*);` Baris ini mendeklarasikan prototipe fungsi untuk nama fungsi `split_string` yang mengambil pointer ke karakter sebagai argumen dan mengembalikan pointer ke pointer ke karakter.
- Baris (11) `int parse_int(char*);` Baris ini mendeklarasikan prototipe fungsi untuk nama fungsi `parse_int` yang menggunakan pointer ke karakter sebagai argumen dan mengembalikan bilangan bulat.
- Baris (13) `int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {` Baris ini mendefinisikan nama fungsi `twoStacks` yang mengambil beberapa parameter: `maxSum` (bilangan bulat mewakili jumlah maksimum yang diizinkan), `a_count` (bilangan bulat mewakili jumlah elemen dalam array `a`), `a` (pointer ke array bilangan bulat `a`), `b_count` (bilangan bulat mewakili jumlah elemen dalam array `b`), dan `b` (penunjuk ke array integer `b`).
- Baris (14) `int max = 0;` Baris ini mendeklarasikan dan menginisialisasi variabel integer `max` dengan nilai 0.
- Baris (15) `int sum = 0;` Baris ini mendeklarasikan dan menginisialisasi variabel integer `sum` dengan nilai 0.
- Baris (16) `int items = 0;` Baris ini mendeklarasikan dan menginisialisasi variabel integer `items` dengan nilai 0.
- Baris (17) `int apos = 0, bpos = 0;` Baris ini mendeklarasikan dan menginisialisasi dua variabel integer `apos` dan `bpos` bernilai 0.
- Baris (18) `while (sum <= maxSum && apos < a_count) {` Baris ini memulai perulangan `while` yang berlanjut selama `sum` kurang dari atau sama dengan `maxSum` dan `apos` kurang dari `a_count`.
- Baris (19) `if (sum + a[apos] > maxSum)` Baris ini memeriksa apakah menambahkan elemen saat ini ke `sum` melebihi `maxSum`.
- Baris (20) `break;` Baris ini keluar dari perulangan `while` jika kondisi pada baris sebelumnya benar.
- Baris (21) `sum += a[apos];` Baris ini menambahkan elemen saat ini ke `sum`.
- Baris (22) `apos++;` Baris ini menambah nilai `apos` sebesar 1.
- Baris (23) `items++;` Baris ini menambah nilai `items` sebesar 1.
- Baris (25) `while (sum <= maxSum && bpos < b_count) {` Baris ini memulai perulangan `while` yang berlanjut selama `sum` kurang dari atau sama dengan `maxSum` dan `bpos` kurang dari `b_count`.

- Baris (26) `if (sum + b[bpos] > maxSum)` Baris ini memeriksa apakah menambahkan elemen saat ini ke `sum` melebihi `maxSum`.
- Baris (27) `break;` Baris ini keluar dari perulangan `while` jika kondisi pada baris sebelumnya benar.
- Baris (28) `sum += b[bpos];` Baris ini menambahkan elemen saat ini ke `sum`.
- Baris (29) `bpos++;` Baris ini menambah nilai `bpos` sebesar 1.
- Baris (30) `items++;` Baris ini menambah nilai `items` sebesar 1.
- Baris (32) `max = items;` Baris ini memberikan nilai `items` ke `max`.
- Baris (33) `while (1) {` Baris ini memulai perulangan `while` yang tak terbatas.
- Baris (34) `if (apos <= 0)` Baris ini memeriksa apakah `apos` kurang dari atau sama dengan 0.
- Baris (35) `break;` Baris ini keluar dari perulangan `while` jika kondisi pada baris sebelumnya benar.
- Baris (36) `apos--;` Baris ini mengurangi nilai `apos` sebesar 1.
- Baris (37) `sum -= a[apos];` Baris ini mengurangi elemen saat ini dari `sum`.
- Baris (38) `items--;` Baris ini mengurangi nilai `items` sebesar 1.
- Baris (39) `while (sum <= maxSum && bpos < b_count) {` Baris ini memulai perulangan `while` yang berlanjut selama `sum` kurang dari atau sama dengan `maxSum` dan `bpos` kurang dari `b_count`.
- Baris (40) `if (sum + b[bpos] > maxSum)` Baris ini memeriksa apakah menambahkan elemen saat ini ke `sum` melebihi `maxSum`.
- Baris (41) `break;` Baris ini keluar dari perulangan `while` jika kondisi pada baris sebelumnya benar.
- Baris (42) `sum += b[bpos];` Baris ini menambahkan elemen saat ini ke `sum`.
- Baris (43) `bpos++;` Baris ini menambah nilai `bpos` sebesar 1.
- Baris (44) `items++;` Baris ini menambah nilai `items` sebesar 1.
- Baris (46) `if (items > max)` Baris ini memeriksa apakah nilai `items` lebih besar dari `max`.
- Baris (47) `max = items;` Baris ini memberikan nilai `items` ke `max`.
- Baris (48) `if (bpos == b_count)` Baris ini memeriksa apakah `bpos` sama dengan `b_count`.
- Baris (49) `break;` Baris ini keluar dari perulangan `while` jika kondisi pada baris sebelumnya benar.
- Baris (51) `return max;` Baris ini mengembalikan nilai `max`.
- Baris (52) `}` Baris ini menutup fungsi.

- Baris (54) `int main() {` Baris ini mendefinisikan fungsi utama, yang merupakan titik masuk program.
- Baris (55) `int g;` Baris ini mendeklarasikan variabel integer `g`.
- Baris (56) `scanf("%d", &g);` Baris ini membaca nilai integer dari pengguna dan menyimpannya dalam variabel `g`.
- Baris (57) `for (int a0 = 0; a0 < g; a0++) {` Baris ini memulai perulangan `for` yang berulang `g` kali. Variabel loop `a0` diinisialisasi ke 0 dan bertambah 1 di setiap iterasi.
- Baris (58) `int n;` Baris ini mendeklarasikan variabel integer `n`.
- Baris (59) `int m;` Baris ini mendeklarasikan variabel integer `m`.
- Baris (60) `int x;` Baris ini mendeklarasikan variabel integer `x`.
- Baris (61) `scanf("%d %d %d", &n, &m, &x);` Baris ini membaca tiga nilai integer dari pengguna dan menyimpannya dalam variabel `n`, `m`, dan `x`.
- Baris (62) `int* a = malloc(sizeof(int) * n);` Baris ini secara dinamis mengalokasikan memori untuk array integer dengan ukuran `n`.
- Baris (63) `for (int a_i = 0; a_i < n; a_i++) {` Baris ini memulai perulangan `for` yang berulang `n` kali. Variabel loop `a_i` diinisialisasi ke 0 dan bertambah 1 di setiap iterasi.
- Baris (64) `scanf("%d", &a[a_i]);` Baris ini membaca nilai integer dari pengguna dan menyimpannya dalam array di indeks `a_i`.
- Baris (66) `int* b = malloc(sizeof(int) * m);` Baris ini secara dinamis mengalokasikan memori untuk array integer dengan ukuran `m`.
- Baris (67) `for (int b_i = 0; b_i < m; b_i++) {` Baris ini memulai perulangan `for` yang berulang `m` kali. Variabel loop `b_i` diinisialisasi ke 0 dan bertambah 1 di setiap iterasi.
- Baris (68) `scanf("%d", &b[b_i]);` Baris ini membaca nilai integer dari pengguna dan menyimpannya dalam array di indeks `b_i`.
- Baris (71) `int result = twoStacks(x, n, a, m, b);` Baris ini memanggil `twoStacks` fungsi dengan argumen yang diberikan dan memberikan nilai yang dikembalikan ke variabel `result`.
- Baris (72) `printf("%d\n", result);` Baris ini mencetak nilai `result` diikuti dengan karakter baris baru.
- Baris (73) `free(a);` Baris ini membebaskan memori yang dialokasikan untuk array.
- Baris (74) `free(b);` Baris ini membebaskan memori yang dialokasikan untuk array.
- Baris (75) `}` Baris ini menutup perulangan `for`.
- Baris (76) `return 0;` Baris ini menunjukkan keberhasilan penghentian program.

- Baris (79) `char* readline()` { Baris ini mendefinisikan `readline` fungsi, yang mengembalikan pointer ke karakter.
- Baris (80) `size_t alloc_length = 1024;` Baris ini mendeklarasikan dan menginisialisasi variabel `alloc_length` bertipe `size_t` dengan nilai 1024, mewakili panjang awal yang dialokasikan untuk data masukan.
- Baris (81) `size_t data_length = 0;` Baris ini mendeklarasikan dan menginisialisasi variabel `data_length` bertipe `size_t` dengan nilai 0, yang mewakili panjang data masukan saat ini.
- Baris (82) `char* data = malloc(alloc_length);` Baris ini secara dinamis mengalokasikan memori untuk array karakter `data` dengan ukuran awal `alloc_length`.
- Baris (83) `while (1)` { Baris ini memulai perulangan `while` tanpa batas untuk membaca masukan baris demi baris.
- Baris (84) `char* cursor = data + data_length;` Baris ini menginisialisasi pointer `cursor` ke posisi saat ini dalam data array dimana input berikutnya akan disimpan.
- Baris (85) `char* line = fgets(cursor, alloc_length - data_length, stdin);` Baris ini membaca baris input dari input standar dan menyimpannya di buffer `line`, memastikan tidak melebihi ruang yang tersisa dalam data array.
- Baris (86 – 87) `if (!line) break;` Baris ini keluar dari loop jika tidak ada lagi input untuk dibaca (akhir file atau kesalahan).
- Baris (88) `data_length += strlen(cursor);` Baris ini memperbarui `data_length` dengan menambahkan panjang input yang baru dibaca.
- Baris (89 - 90) `if (data_length < alloc_length - 1 || data[data_length - 1] == '\n') break;` Baris ini memeriksa apakah panjang data input kurang dari panjang yang dialokasikan atau apakah karakter terakhir adalah baris baru, dan keluar dari loop jika benar.
- Baris (91) `alloc_length <<= 1;` Baris ini menggandakan panjang yang dialokasikan untuk data masukan.
- Baris (92) `data = realloc(data, alloc_length);` Baris ini mengalokasikan kembali memori untuk data array dengan `alloc_length`.
- Baris (93 – 95) `if (!data) { data = '\0'; break; }` Baris ini menangani kasus di mana realokasi memori gagal dengan menyetel `data` ke null dan keluar dari loop.
- Baris (98 – 109) Kode berikutnya menangani skenario di mana karakter terakhir dari data masukan adalah baris baru atau bukan, dan mengalokasikan kembali memori sesuai untuk memastikan data masukan diakhiri dengan benar dengan karakter nol.
- Baris (110) `return data;` Baris ini mengembalikan pointer ke data input.

- Baris (113) `char* ltrim(char* str)` { Baris ini mendefinisikan fungsi bernama `ltrim` yang mengambil `char*nama` (penunjuk karakter) `str` sebagai argumen dan mengembalikan `char*` sebagai hasilnya.
- Baris (114-115) `if (!str) return '\0'`; Baris ini memeriksa apakah inputnya `str` adalah penunjuk nol. Jika ya, fungsi akan mengembalikan karakter nol (`'\0'`).
- Baris (116-117) `if (!*str) return str`; Baris ini memeriksa apakah karakter pertama `str` adalah karakter nol (`'\0'`). Jika ya, fungsinya akan kembali `str` sendiri.
- Baris (118-119) `while (*str != '\0' && isspace(*str)) str++`; Baris ini memulai perulangan `while` yang berlanjut selama karakter saat ini `str` bukan karakter nol (`'\0'`) dan merupakan karakter spasi putih (sebagaimana ditentukan oleh `isspace` fungsi dari pustaka standar C). Di dalam loop, `str` penunjuk bertambah untuk berpindah ke karakter berikutnya.
- Baris (120-121) `return str; }` Baris ini mengembalikan `str` penunjuk yang dimodifikasi, yang sekarang menunjuk ke karakter non-spasi pertama dalam string masukan.
- Baris (123) `char* rtrim(char* str)` { Baris ini mendefinisikan fungsi bernama `rtrim` yang mengambil `char*nama` (penunjuk karakter) `str` sebagai argumen dan mengembalikan `char*` sebagai hasilnya.
- Baris (124-125) `if (!str) return '\0'`; Baris ini memeriksa apakah inputnya `str` adalah penunjuk nol. Jika ya, fungsi akan mengembalikan karakter nol (`'\0'`).
- Baris (126-127) `if (!*str) return str`; Baris ini memeriksa apakah karakter pertama `str` adalah karakter nol (`'\0'`). Jika ya, fungsinya akan kembali `str` sendiri.
- Baris (128) `char* end = str + strlen(str) - 1`; Baris ini mendeklarasikan `char*nama` variabel `end` dan menetapkan alamat karakter terakhir dalam `str` string dengan menambahkan panjang `str` ke alamat awal dan mengurangi 1.
- Baris (129-130) `while (end >= str && isspace(*end)) end--`; Baris ini memulai perulangan `while` yang berlanjut selama `end` penunjuk lebih besar atau sama dengan `str` penunjuk dan karakter saat ini yang ditunjuk oleh `end` adalah karakter spasi putih (sebagaimana ditentukan oleh fungsi `isspace` dari pustaka standar C). Di dalam loop, `end` penunjuk dikurangi untuk berpindah ke karakter sebelumnya.
- Baris (131) `*(end + 1) = '\0'`; Baris ini menyetel karakter segera setelah karakter non-spasi terakhir menjadi karakter nol (`'\0'`), yang secara efektif memotong string pada titik tersebut.

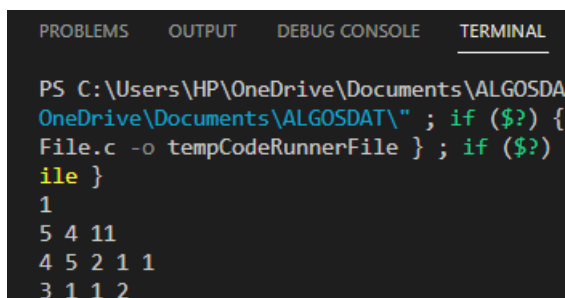
- Baris (132-133) `return str; }` Baris ini mengembalikan `str` yang dimodifikasi, yang sekarang menunjuk ke string yang dipangkas dengan karakter spasi tambahan yang dihapus.
- Baris (135) `char** split_string(char* str) {` Baris ini mendefinisikan fungsi bernama `split_string` yang mengambil `char* nama` (penunjuk karakter) `str` sebagai argumen dan mengembalikan `char**` (penunjuk ke penunjuk ke karakter) sebagai hasilnya.
- Baris (136) `char** splits = NULL;` Baris ini mendeklarasikan `char** nama` variabel `splits` dan menginisiasinya menjadi `NULL`. Variabel ini akan menyimpan array string yang dipisahkan.
- Baris (137) `char* token = strtok(str, " ");` Baris ini menggunakan `strtok` fungsi dari pustaka standar C untuk memberi token pada `str` menggunakan spasi (" ") sebagai pembatas. Panggilan pertama untuk `strtok` mengembalikan pointer ke token pertama yang ditemukan di `str`, dan panggilan berikutnya dengan `NULL` sebagai argumen pertama melanjutkan tokenisasi string hingga tidak ada lagi token. Token yang dikembalikan disimpan dalam `token` variabel.
- Baris (138) `int spaces = 0;` Baris ini mendeklarasikan nama variabel integer `spaces` dan menginisiasinya ke 0. Variabel ini akan melacak jumlah token yang ditemukan dalam string.
- Baris (139) `while (token) {` Baris ini memulai perulangan `while` yang berlanjut selama variabelnya `token` bukan `NULL`, yang menunjukkan bahwa ada lebih banyak token yang harus diproses.
- Baris (140) `splits = realloc(splits, sizeof(char*) * ++spaces);` Baris ini menggunakan `realloc` fungsi dari pustaka standar C untuk mengalokasikan memori secara dinamis untuk `splits` array. Ukuran memori yang dialokasikan bertambah dengan `sizeof(char*)` dikalikan dengan nilai pertambahan `spaces`. Hal ini memastikan bahwa memori yang cukup dialokasikan untuk menyimpan token tambahan.
- Baris (141-142) `if (!splits) return splits;` Baris ini memeriksa apakah penggunaan alokasi memori `realloc` berhasil. Jika `splits` penunjuknya adalah `NULL`, yang menunjukkan kegagalan dalam alokasi memori, fungsi akan segera mengembalikan `splits` penunjuk tersebut.
- Baris (143) `splits[spaces - 1] = token;` Baris ini menetapkan token saat ini ke elemen terakhir array `splits`, menggunakan `spaces` variabel sebagai indeks.
- Baris (144-145) `token = strtok(NULL, " "); }` Baris ini memanggil `strtok` lagi with `NULL` sebagai argumen pertama untuk melanjutkan tokenisasi string. Kali ini,

fungsinya menggunakan pembatas yang sama (" ") seperti sebelumnya. Token yang dikembalikan disimpan dalam tokenvariabel, dan perulangan berlanjut hingga tidak ada lagi token.

- Baris (146-147) `return splits; }` Baris ini mengembalikan `splits`penunjuk, yang sekarang menunjuk ke larik string yang diperoleh dengan memisahkan string masukan berdasarkan pembatas.
- Baris (149) `int parse_int(char* str) {` Baris ini mendefinisikan fungsi bernama `parse_int` yang mengambil `char*`(penunjuk karakter) bernama `str` sebagai argumen dan mengembalikan `int` sebagai hasilnya.
- Baris (150) `char* endptr;` Baris ini mendeklarasikan `char*` variabel bernama `endptr`. Variabel ini akan digunakan untuk menyimpan alamat karakter tidak valid pertama yang ditemukan selama konversi.
- Baris (151) `int value = strtol(str, &endptr, 10);` Baris ini menggunakan `strtol` fungsi dari perpustakaan standar C untuk mengubah string input menjadi bilangan bulat. Fungsi ini `strtol` memerlukan tiga argumen: string yang akan dikonversi (`str`), penunjuk ke `char*` variabel (`endptr`) yang akan menyimpan alamat karakter pertama yang tidak valid, dan basis (`10`) untuk konversi. Nilai integer yang dikonversi disimpan dalam `value` variabel.
- Baris (152-154) `if (endptr == str || *endptr != '\0') { exit(EXIT_FAILURE); }` Baris ini memeriksa apakah `endptr` sama dengan `str` atau apakah karakter yang ditunjuk `endptr` bukan karakter nol (`'\0'`). Jika salah satu dari kondisi ini benar, berarti konversi tidak berhasil, dan program keluar dengan status kegagalan menggunakan fungsi `exit` dari pustaka standar C.
- Baris (155-156) `return value; }` Baris ini mengembalikan nilai integer yang dikonversi.

SS Input

Sesuai permintaan di soal.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\HP\OneDrive\Documents\ALGOSDATA> g++ OneDrive\Documents\ALGOSDATA\1.c -o tempCodeRunnerFile && if ($?) { .\tempCodeRunnerFile }
1
5 4 11
4 5 2 1 1
3 1 1 2
```

SS Output

5

PS C:\Users\HP\OneDrive\Documents\ALGOSDAT>

Source Code

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <ctype.h>
5
6 char* readline();
7 char* ltrim(char*);
8 char* rtrim(char*);
9 char** split_string(char*);
10
11 int parse_int(char*);
12
13 int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
14     int max = 0;
15     int sum = 0;
16     int items = 0;
17     int apos = 0, bpos = 0;
18     while (sum <= maxSum && apos < a_count) {
19         if (sum + a[apos] > maxSum)
20             break;
21         sum += a[apos];
22         apos++;
23         items++;
24     }
25     while (sum <= maxSum && bpos < b_count) {
26         if (sum + b[bpos] > maxSum)
27             break;
28         sum += b[bpos];
29         bpos++;
30         items++;
31     }
32     max = items;
33     while (1) {
34         if (apos < 0)
35             break;
36         apos--;
37         sum -= a[apos];
38         items--;
39         while (sum <= maxSum && bpos < b_count) {
40             if (sum + b[bpos] > maxSum)
41                 break;
42             sum += b[bpos];
43             bpos++;
44             items++;
45         }
46         if (items > max)
47             max = items;
48         if (bpos == b_count)
49             break;
50     }
51     return max;
52 }
53
54 int main() {
55     int g;
56     scanf("%d", &g);
57     for (int ab = 0; ab < g; ab++) {
58         int n;
59         int m;
60         int k;
61         scanf("%d %d %d", &n, &m, &k);
62         int* a = malloc(sizeof(int) * n);
63         for (int i = 0; i < n; i++) {
64             scanf("%d", &a[i]);
65         }
66         int* b = malloc(sizeof(int) * m);
67         for (int i = 0; i < m; i++) {
68             scanf("%d", &b[i]);
69         }
70         // your code goes here
71         int result = twoStacks(k, n, a, m, b);
72         printf("%d\n", result);
73         free(a);
74         free(b);
75     }
76     return 0;
77 }
78
79 char* readline() {
80     size_t alloc_length = 1024;
81     size_t data_length = 0;
82     char* data = malloc(alloc_length);
83     while (1) {
84         char* cursor = data + data_length;
85         char* line = fgets(cursor, alloc_length - data_length, stdin);
86         if (!line)
87             break;
88         data_length += strlen(cursor);
89         if (data_length < alloc_length - 1 || data[data_length - 1] != '\n')
90             break;
91         alloc_length <= 1;
92         data = realloc(data, alloc_length);
93         if (!data) {
94             data = '\0';
95             break;
96         }
97     }
98     if (data[data_length - 1] == '\n') {
99         data[data_length - 1] = '\0';
100         data = realloc(data, data_length);
101         if (!data)
102             data = '\0';
103     } else {
104         data = realloc(data, data_length + 1);
105         if (!data)
106             data = '\0';
107     }
108     else
109         data[data_length] = '\0';
110 }
111 return data;
112 }
113
114 char* ltrim(char* str) {
115     if (!str)
116         return '\0';
117     if (!*str)
118         return str;
119     while (*str != '\0' && isspace(*str))
120         str++;
121     return str;
122 }
123
124 char* rtrim(char* str) {
125     if (!str)
126         return '\0';
127     if (!*str)
128         return str;
129     char* end = str + strlen(str) - 1;
130     while (end >= str && isspace(*end))
131         end--;
132     *(end + 1) = '\0';
133     return str;
134 }
135
136 char** split_string(char* str) {
137     char** splits = NULL;
138     char* token = strtok(str, " ");
139     int spaces = 0;
140     while (token) {
141         splits = realloc(splits, sizeof(char*) * ++spaces);
142         if (!splits)
143             return splits;
144         splits[spaces - 1] = token;
145         token = strtok(NULL, " ");
146     }
147     return splits;
148 }
149
150 int parse_int(char* str) {
151     char* endptr;
152     int value = strtol(str, &endptr, 10);
153     if (endptr == str || *endptr != '\0') {
154         exit(EXIT_FAILURE);
155     }
156     return value;
157 }
```