

Arka dwi indrastata

1203230017

IF-03-02

TUGAS OTH

Soal no.1

#source code

```
#include <stdio.h>

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

int main()
{
    struct element 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;
```

```

16.alphabet = 'T';

17.link = NULL;
17.alphabet = 'N';

18.link = NULL;
18.alphabet = 'O';

19.link = NULL;
19.alphabet = 'R';

14.link = &17;
17.link = &11;
11.link = &18;
18.link = &19;
19.link = &12;
12.link = &13;
13.link = &16;
16.link = &14;

printf("%c", 14.alphabet);
printf("%c", 14.link->alphabet);
printf("%c", 14.link->link->alphabet);
printf("%c", 14.link->link->link->alphabet);
printf("%c", 14.link->link->link->link->alphabet);
printf("%c", 14.link->link->link->link->link->alphabet);
printf("%c", 14.link->link->link->link->link->link->alphabet);
printf("%c", 14.link->link->link->link->link->link->link->alphabet);

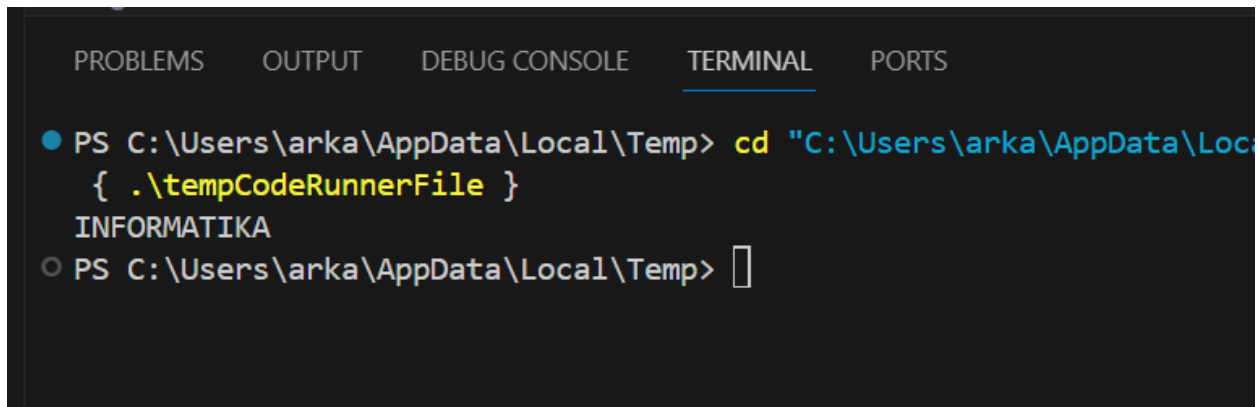
14.link = &15;
15.link = &13;

printf("%c", 14.link->alphabet);
printf("%c", 14.link->link->alphabet);

return 0;
}

```

#OUTPUT

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. The terminal shows a PowerShell prompt 'PS C:\Users\arka\AppData\Local\Temp>' followed by the command 'cd "C:\Users\arka\AppData\Local\Temp\{ .\tempCodeRunnerFile }"'. Below this, the text 'INFORMATIKA' is displayed. The prompt returns to 'PS C:\Users\arka\AppData\Local\Temp>' with a cursor at the end.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● PS C:\Users\arka\AppData\Local\Temp> cd "C:\Users\arka\AppData\Local\Temp\{ .\tempCodeRunnerFile }"
INFORMATIKA
○ PS C:\Users\arka\AppData\Local\Temp> █
```

#Penjelasan

1. Pertama kita deklarasikan terlebih dahulu struktur **element** yang akan digunakan untuk mempresentasikan setiap node dalam linked list, struktur ini memiliki 2 anggota yaitu: **link** dan **alphabet**
2. Fungsi main(): Ini adalah fungsi utama yang akan dieksekusi saat program dijalankan.
3. Inisialisasi element Node: Kemudian, kita akan mendeklarasikan dan menginisialisasi sembilan node (l1 hingga l9)
4. Lalu kita akan menghubungkan node node hingga membentuk linked list
5. Setelah itu kita akan mengprint untuk mencetak isi linked list menggunakan printf
6. Lalu kita mengubah struktur linked list dengan mengubah pointer node l4 untuk menunjuk ke node l5, dan node l5 untuk menunjuk ke node l3.
7. Lalu kita mencetak Kembali isi linked list setelah dilakukan perubahan
8. Program diakhiri dengan return 0 menandakan akhir program.

Soal no 2

Source code

1.<assert.h>: Mendefinisikan makro-makro untuk melakukan asersi pada program. Fungsi assert digunakan untuk memeriksa kondisi yang seharusnya benar dalam program. Jika kondisi tersebut salah, program akan menghasilkan kesalahan runtime.**<ctype.h>:** Mendefinisikan fungsi-fungsi untuk mengecek dan memanipulasi karakter. Contohnya isdigit, isalpha, toupper, tolower, dan sebagainya.**<limits.h>:** Mendefinisikan batasan-batasan numerik untuk tipe data primitif, seperti INT_MAX, INT_MIN, CHAR_BIT, dan lain-lain.**<math.h>:** Mendefinisikan fungsi-fungsi matematika, seperti fungsi trigonometri, fungsi logaritma, fungsi pangkat, dan lainnya.**<stdbool.h>:** Mendefinisikan tipe data boolean (true dan false) serta nilai-nilai yang terkait, seperti true, false, dan bool.**<stddef.h>:** Mendefinisikan berbagai tipe dan makro, seperti NULL, size_t, offsetof, dan lain-lain.**<stdint.h>:** Mendefinisikan tipe data integer dengan ukuran yang tepat (int8_t, int16_t, int32_t, dll) serta konstanta-konstanta yang berkaitan dengan ukuran tipe data, seperti INT8_MAX, INT16_MAX, INT32_MAX, dll.**<stdio.h>:** Mendefinisikan fungsi-fungsi input-output standar, seperti printf, scanf, fopen, fclose, dan lainnya.**<stdlib.h>:** Mendefinisikan fungsi-fungsi dasar dalam pemrograman C, seperti alokasi memori (malloc, calloc, realloc), konversi string ke angka (atoi, atol, atof), fungsi pengurutan (qsort), dan sebagainya.**<string.h>:** Mendefinisikan fungsi-fungsi pemrosesan string, seperti strlen, strcpy, strcat, strcmp, dan lain-lain.

2. **char *getline():** Membaca satu baris teks dari stdin.**char *ltrim(char *):** Menghapus spasi di awal (kiri) **string.char *rtrim(char *):** Menghapus spasi di akhir (kanan) **string.char **split_string(char *):** Membagi string menjadi potongan-potongan berdasarkan delimiter tertentu.

```
3. int parse_int(char *);

int twoStacks(int maxSum, int a_count, int *a, int b_count, int *b)
{
    int i = 0, j = 0, sum = 0, count = 0;
    while (i < a_count && sum + a[i] <= maxSum)
    {
        sum += a[i];
        i++;
    }
    count = i;
    while (j < b_count && i >= 0)
    {
        sum += b[j];
        j++;
        while (sum > maxSum && i > 0)
```

```

    {
        i--;
        sum -= a[i];
    }
    if (sum <= maxSum && i + j > count)
    {
        count = i + j;
    }
}
return count;
}

```

Program ini merupakan implementasi algoritma untuk menyelesaikan permasalahan dua tumpukan (two stacks) dengan batasan tertentu. Fungsi `parse_int` digunakan untuk mengonversi string menjadi bilangan bulat. Ini berguna saat membaca input dari pengguna yang berupa string dan membutuhkan konversi ke tipe data bilangan bulat.

Fungsi `twoStacks` adalah inti dari program. Fungsi ini menerima lima parameter: `maxSum` (nilai maksimum yang tidak boleh dilewati saat menambahkan angka dari kedua tumpukan) dan masing-masing `a_count`, `a`, `b_count`, dan `b`, yang mewakili jumlah elemen dalam tumpukan pertama (`a`), elemen-elemen dalam tumpukan pertama (`a`), jumlah elemen dalam tumpukan kedua (`b`), dan elemen-elemen dalam tumpukan kedua (`b`).

- Iterasi pertama dilakukan pada tumpukan pertama (`a`) untuk menambahkan sebanyak mungkin angka ke dalam `sum` tanpa melebihi `maxSum`.
- Setelah mencapai batas maksimum atau selesai menambahkan angka, dilakukan iterasi pada tumpukan kedua (`b`). Saat iterasi, angka dari tumpukan kedua (`b`) akan ditambahkan ke `sum`.
- Jika jumlah `sum` melebihi `maxSum`, maka angka dari tumpukan pertama (`a`) akan dihapus satu per satu sampai `sum` tidak lagi melebihi `maxSum` atau elemen dalam tumpukan pertama habis.
- Pada setiap langkah, jumlah angka yang berhasil ditambahkan dari kedua tumpukan akan diperbarui. Jika jumlah angka yang berhasil ditambahkan lebih besar dari nilai sebelumnya, maka nilai tersebut akan diupdate.

```

4. int main()
{
    FILE *fptr = fopen(getenv("OUTPUT_PATH"), "w");

    int g = parse_int(ltrim(rtrim(readline())));

    for (int g_itr = 0; g_itr < g; g_itr++)
    {

```

```

char **first_multiple_input = split_string(rtrim(readline()));

int n = parse_int(*(first_multiple_input + 0));

int m = parse_int(*(first_multiple_input + 1));

int maxSum = parse_int(*(first_multiple_input + 2));

char **a_temp = split_string(rtrim(readline()));

int *a = malloc(n * sizeof(int));

for (int i = 0; i < n; i++)
{
    int a_item = parse_int(*(a_temp + i));

    *(a + i) = a_item;
}

char **b_temp = split_string(rtrim(readline()));

int *b = malloc(m * sizeof(int));

for (int i = 0; i < m; i++)
{
    int b_item = parse_int(*(b_temp + i));

    *(b + i) = b_item;
}

int result = twoStacks(maxSum, n, a, m, b);

fprintf(fp_ptr, "%d\n", result);
}

fclose(fp_ptr);

return 0;
}

```

Kode tersebut adalah implementasi dari fungsi main() yang digunakan untuk menjalankan program. Berikut adalah penjelasan singkatnya:

Program membuka sebuah file dengan nama yang disimpan dalam variabel lingkungan OUTPUT_PATH untuk ditulis (w). File ini akan digunakan untuk menyimpan output program.

Program membaca nilai g dari input, yang menunjukkan jumlah kasus uji yang akan dijalankan.

Program melakukan loop for sebanyak g kali untuk setiap kasus uji:

- a. Membaca input yang terdiri dari tiga nilai: n, m, dan maxSum. n adalah jumlah elemen dalam tumpukan pertama, m adalah jumlah elemen dalam tumpukan kedua, dan maxSum adalah nilai maksimum yang tidak boleh dilewati saat menambahkan angka dari kedua tumpukan.
- b. Membaca elemen-elemen tumpukan pertama (a) dan tumpukan kedua (b) dari input.
- c. Memanggil fungsi twoStacks untuk menghitung jumlah maksimum angka yang dapat diambil dari kedua tumpukan tanpa melebihi maxSum.
- d. Menulis hasil perhitungan ke dalam file output.

Setelah selesai semua kasus uji, file output ditutup dan program selesai.

```
5. char *getline()
{
    size_t alloc_length = 1024;
    size_t data_length = 0;

    char *data = malloc(alloc_length);

    while (true)
    {
        char *cursor = data + data_length;
        char *line = fgets(cursor, alloc_length - data_length, stdin);

        if (!line)
        {
            break;
        }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_length - 1] == '\n')
        {
            break;
        }

        alloc_length <<= 1;
```

```

        data = realloc(data, alloc_length);

        if (!data)
        {
            data = '\0';

            break;
        }
    }

    if (data[data_length - 1] == '\n')
    {
        data[data_length - 1] = '\0';

        data = realloc(data, data_length);

        if (!data)
        {
            data = '\0';
        }
    }
    else
    {
        data = realloc(data, data_length + 1);

        if (!data)
        {
            data = '\0';
        }
        else
        {
            data[data_length] = '\0';
        }
    }

    return data;
}

```

```

char *ltrim(char *str)
{
    if (!str)
    {
        return '\0';
    }
}

```



```
    if (!*str)
    {
        return str;
    }

    while (*str != '\0' && isspace(*str))
    {
        str++;
    }

    return str;
}
```

```
char *rtrim(char *str)
{
    if (!str)
    {
        return '\0';
    }

    if (!*str)
    {
        return str;
    }

    char *end = str + strlen(str) - 1;

    while (end >= str && isspace(*end))
    {
        end--;
    }

    *(end + 1) = '\0';

    return str;
}
```

```
char **split_string(char *str)
{
    char **splits = NULL;
    char *token = strtok(str, " ");

    int spaces = 0;
```

```

while (token)
{
    splits = realloc(splits, sizeof(char *) * ++spaces);

    if (!splits)
    {
        return splits;
    }

    splits[spaces - 1] = token;

    token = strtok(NULL, " ");
}

return splits;
}

int parse_int(char *str)
{
    char *endptr;
    int value = strtol(str, &endptr, 10);

    if (endptr == str || *endptr != '\0')
    {
        exit(EXIT_FAILURE);
    }

    return value;
}

```

Kode tersebut merupakan implementasi dari beberapa fungsi bantuan yang digunakan dalam program. Berikut adalah penjelasan singkatnya:

`getline()`: Membaca satu baris teks dari input standar (stdin). Ini dilakukan dengan alokasi memori dinamis untuk menyimpan data, kemudian membaca satu baris teks menggunakan `fgets()`. Jika memori yang dialokasikan tidak cukup, ukuran alokasi diperbesar secara dinamis menggunakan `realloc()`. Fungsi ini juga memastikan bahwa data yang dibaca tidak memiliki karakter newline (`\n`) di akhirnya.

`ltrim()`: Menghapus spasi di awal (kiri) string. Fungsi ini menggeser pointer ke depan hingga menemukan karakter non-spasi.

`rtrim()`: Menghapus spasi di akhir (kanan) string. Fungsi ini memeriksa karakter di akhir string dan menggantinya dengan `\0` jika karakter tersebut adalah spasi.

`split_string()`: Membagi string menjadi potongan-potongan berdasarkan delimiter tertentu (dalam kasus ini, spasi). Fungsi ini menggunakan `strtok()` untuk membagi string menjadi token-token dan alokasi memori dinamis untuk menyimpan token-token tersebut dalam array.

`parse_int()`: Mengonversi string menjadi bilangan bulat. Fungsi ini menggunakan `strtol()` untuk mengonversi string menjadi bilangan bulat. Jika string tidak valid atau tidak dapat dikonversi, program akan berhenti dengan pemanggilan `exit(EXIT_FAILURE)`.

```
{ .\tempCodeRunnerFile }  
1  
5 4 11  
4 5 2 1 1  
3 1 1 2  
PS C:\Users\arka\AppData\Local\Temp
```

Hackerank

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)

[Download](#)

1	1
2	5 4 10
3	4 2 4 6 1
4	2 1 8 5

Your Output (stdout)

1	4
---	---

Expected Output

[Download](#)

1	4
---	---