LAPRAK TUGAS OTH STRUCK DAN STACK

Nama : Ilham Prakosa

Nim : 1203230027

1.Source Code

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

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

int main() {
    // Deklarasi node-node
    struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
    struct Node *link, *l3ptr;

    // Inisialisasi node-node dengan menggunakan potongan kode soal
    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";

    // Mengatur koneksi antar node sesuai dengan urutan yang diinginkan
    l7.link = &l1;// Menyambungkan ke l1
    l1.link = &l8;// Menyambungkan ke l1
    l8.link = &l2;// Menyambungkan ke l1
    l2.link = &l5;// Menyambungkan ke l1
    l5.link = &l3;// Menyambungkan ke l1
    l3.link = &l6;// Menyambungkan ke l1
    l6.link = &l9;
    l9.link = &l4;
    l4.link = &l7;

    // Starting point
    l3ptr = &l7;

    // Akses data menggunakan printf
    printf("%s", l3.link->link->link->alphabet);// Menampilkan huruf I
    printf("%s", l3.link->link->link->link->alphabet);// Menampilkan huruf N
    printf("%s", l3.link->link->link->link->link->alphabet);// Menampilkan huruf
F
    printf("%s", l3.link->link->link->link->link->link->alphabet);// Menampilkan
huruf O
    printf("%s", l3.link->link->alphabet);// Menampilkan huruf R
    printf("%s", l3.link->link->link->link->link->link->link->alphabet);//
Menampilkan huruf M
    printf("%s", l3.alphabet);// Menampilkan huruf A
    printf("%s", l3.link->alphabet);// Menampilkan huruf T
    printf("%s", l3.link->link->link->alphabet);// Menampilkan huruf I
    printf("%s", l3.link->link->link->link->link->link->link->link->alphabet);//
Menampilkan huruf K
    printf("%s", l3.alphabet);// Menampilkan huruf A


    return 0;
}
```

Penjelasan:

1. #include <stdio.h>: Ini adalah preprocessor directive yang menyertakan file header stdio.h, yang berisi deklarasi fungsi standar input-output seperti printf() dan scanf().
2. #include <stdlib.h>: Ini adalah preprocessor directive yang menyertakan file header stdlib.h, yang berisi deklarasi fungsi-fungsi standar seperti alokasi memori (malloc(), calloc(), free()).

3. struct Node { char* alphabet; struct Node* link; };: Ini mendefinisikan sebuah struktur Node yang memiliki dua anggota, yaitu alphabet bertipe pointer ke karakter (char*) dan link bertipe pointer ke struktur Node itu sendiri.
4. int main() {: Ini adalah awal dari fungsi main(), yang merupakan titik awal dari eksekusi program.
5. struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;: Ini mendeklarasikan sembilan variabel bertipe struct Node dengan nama l1, l2, ..., l9.
6. struct Node *link, *l3ptr;: Ini mendeklarasikan dua pointer ke struktur Node, yaitu link dan l3ptr.
7. l1.link = NULL; l1.alphabet = "F";: Inisialisasi l1, memberikan link nilai NULL dan alphabet nilai string "F". Ini juga berlaku untuk l2 hingga l9.
8. l7.link = &l1; l1.link = &l8; l8.link = &l2; ...: Ini menyambungkan node-node dalam urutan yang diinginkan.
9. l3ptr = &l7;: l3ptr diatur untuk menunjuk ke node l7, yang akan menjadi titik awal untuk akses data.
10. printf("%s", l3.link->link->link->alphabet);: Ini mencetak nilai dari anggota alphabet dari node yang ditunjuk oleh l3, yang diikuti oleh tiga langkah ke depan ke node selanjutnya.
11. Baris-baris berikutnya pada printf() adalah langkah-langkah yang serupa untuk mencetak huruf dari node-node yang berbeda dalam struktur node, dengan menggunakan operator panah -> untuk mengakses anggota dari node yang ditunjuk oleh pointer.
12. return 0;: Ini menandakan akhir dari fungsi main() dan mengindikasikan bahwa program telah berakhir dengan sukses.

Output :

```
PS C:\TugasMahaDewa\output> & .\'TGSMT2.16.exe'
INFORMATIKA
```

2.

```c
#include <assert.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char* ltrim(char*);
```

```c
char* rtrim(char*);
char** split_string(char*);

int parse_int(char*);

/*
 * Complete the 'twoStacks' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER maxSum
 *  2. INTEGER_ARRAY a
 *  3. INTEGER_ARRAY b
 */

int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
    int a_index = 0, b_index = 0;
    int count = 0, sum = 0;
    // Hitung berapa banyak elemen dari stack a yang dapat diambil
    while (a_index < a_count && sum + a[a_index] <= maxSum) {
        sum += a[a_index];
        a_index++;
        count++;
    }
    int maxCount = count;
    // Coba menambahkan elemen dari stack b
    while (b_index < b_count && a_index >= 0) {
        sum += b[b_index];
        b_index++;
        // Kurangi elemen dari stack a sampai totalnya kurang dari atau sama
dengan maxSum
        while (sum > maxSum && a_index > 0) {
            a_index--;
            sum -= a[a_index];
        }
        // Periksa apakah jumlah elemen saat ini lebih besar dari yang sudah
diperoleh
        if (sum <= maxSum && count < a_index + b_index) {
            count = a_index + b_index;
        }
    }
    return count;
}

int main()
```

```c
{
    // 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(fptr, "%d\n", result);
        printf("%d\n", result);

        free(a);
        free(b);
    }

    // fclose(fptr);
```

```
    return 0;
}

char* readline() {
    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 = NULL;

            break;
        }
    }

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

        data = realloc(data, data_length);

        if (!data) {
            data = NULL;
        }
    } else {
        data = realloc(data, data_length + 1);
```

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

    return data;
}

char* ltrim(char* str) {
    if (!str) {
        return NULL;
    }

    if (!*str) {
        return str;
    }

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

    return str;
}

char* rtrim(char* str) {
    if (!str) {
        return NULL;
    }

    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;
}
```

Penjelasan :

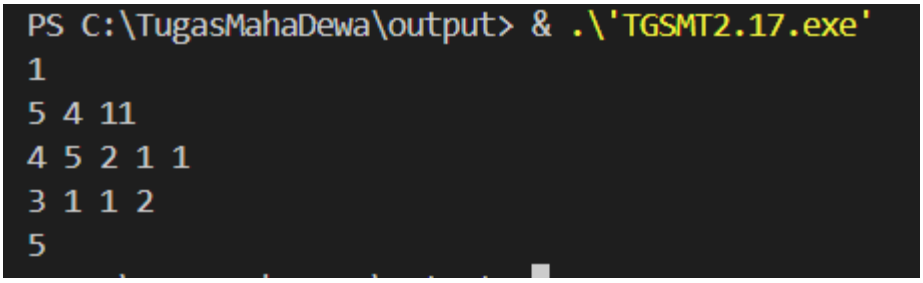
1. #include <assert.h> hingga #include <stdlib.h>: Ini adalah preprocessor directives yang menyertakan file header dari beberapa library standar C yang akan digunakan dalam program. Misalnya, assert.h digunakan untuk melakukan asersi, stdlib.h untuk alokasi memori dan konversi string ke angka.
2. Deklarasi fungsi char* readline();, char* ltrim(char*);, char* rtrim(char*);, char** split_string(char*);, dan int parse_int(char*);: Ini adalah deklarasi prototipe fungsi yang akan digunakan dalam program.

3. Deklarasi fungsi int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b): Ini adalah deklarasi fungsi twoStacks yang diharapkan mengembalikan sebuah nilai integer dan menerima beberapa parameter, seperti maxSum (nilai maksimum yang diizinkan untuk jumlah elemen di dua tumpukan), a_count (jumlah elemen dalam tumpukan a), a (array yang berisi elemen tumpukan a), b_count (jumlah elemen dalam tumpukan b), dan b (array yang berisi elemen tumpukan b).
4. Definisi fungsi int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {...}: Ini adalah definisi dari fungsi twoStacks. Fungsi ini mengimplementasikan algoritma untuk menemukan jumlah maksimum elemen yang dapat diambil dari kedua tumpukan a dan b sedemikian rupa sehingga total nilai elemen yang diambil tidak melebihi maxSum.
5. Fungsi main() {...}: Ini adalah fungsi utama dari program. Program dimulai dari sini.
6. int g = parse_int(ltrim(rtrim(readline())));: Ini membaca input pertama, yaitu jumlah kasus uji (g), menggunakan fungsi readline(), kemudian membersihkan leading dan trailing whitespace menggunakan ltrim() dan rtrim(), dan kemudian mengonversi string tersebut menjadi bilangan bulat menggunakan parse_int().
7. for (int g_itr = 0; g_itr < g; g_itr++) {...}: Ini adalah loop for yang berjalan sebanyak g kali, yang sesuai dengan jumlah kasus uji.
8. char** first_multiple_input = split_string(rtrim(readline()));: Ini membaca tiga nilai pertama dari setiap kasus uji, yaitu n, m, dan maxSum, menggunakan readline() untuk membaca baris input, kemudian membersihkan leading dan trailing whitespace menggunakan rtrim(), dan kemudian memisahkan string input tersebut menjadi array string menggunakan split_string().
9. int result = twoStacks(maxSum, n, a, m, b);: Ini memanggil fungsi twoStacks() dengan parameter yang sesuai, dan menyimpan hasilnya di dalam variabel result.
10. printf("%d\n", result);: Ini mencetak hasil dari setiap kasus uji.
11. free(a); dan free(b);: Ini adalah pembebasan memori yang telah dialokasikan sebelumnya untuk array a dan b dalam setiap kasus uji.
12. char* readline() {...}, char* ltrim(char* str) {...}, char* rtrim(char* str) {...}, char** split_string(char* str) {...}, dan int parse_int(char* str) {...}: Ini adalah implementasi dari fungsi-fungsi bantuan yang digunakan dalam program. readline() digunakan untuk membaca satu baris input, ltrim() dan rtrim() untuk membersihkan leading dan trailing whitespace dari string, split_string() untuk memisahkan string menjadi array string, dan parse_int() untuk mengonversi string menjadi bilangan bulat.

Output :


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
5 4 10
4 2 4 6 1
2 1 8 5
Expected Output
Download
4


Input (stdin)
1
5 4 11
4 5 2 1 1
3 1 1 2
Your Output (stdout)
5


PS C:\TugasMahaDewa\output> & .\'TGSMT2.17.exe'
1
5 4 11
4 5 2 1 1
3 1 1 2
5