Agastya Ferdian Putra Rahardjo

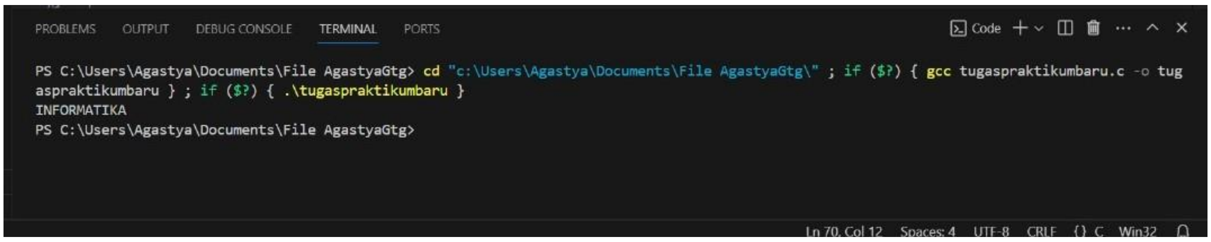
IF-03-01/1203230124

Latihan Soal OTH

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;
}
```

Output:



```
PS C:\Users\Agastya\Documents\File AgastyaGtg> cd "c:\Users\Agastya\Documents\File AgastyaGtg\" ; if ($?) { gcc tugaspraktikumbaru.c -o tug
aspraktikumbaru } ; if ($?) { .\tugaspraktikumbaru }
INFORMATIKA
PS C:\Users\Agastya\Documents\File AgastyaGtg>
```

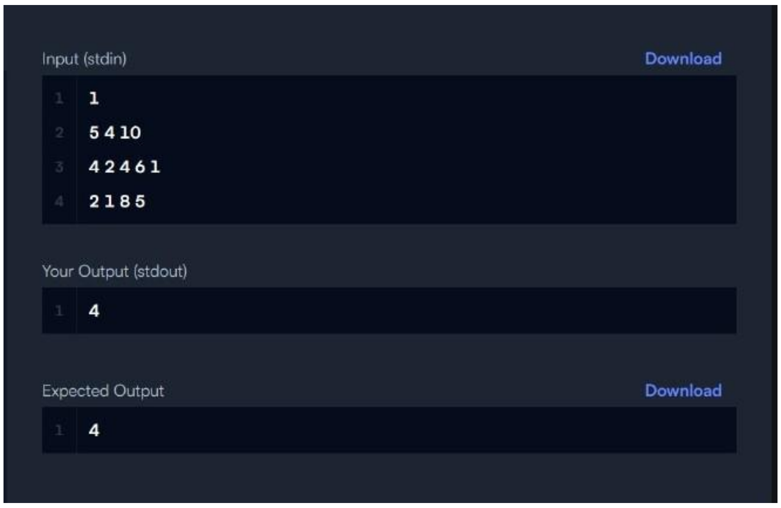
2. Source Code :

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

char* readline();
char* ltrim(char*);
char* rtrim(char*);
char** split_string(char*);
int parse_int(char*);

int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
    int count = 0;
    int sum = 0;
    int idx_a = 0, idx_b = 0;

    while (idx_a < a_count && sum + a[idx_a] <= maxSum) {
        sum += a[idx_a];
        idx_a++;
        count++;
    }

    int max_count = count;

    while (idx_b < b_count && idx_a >= 0) {
        sum += b[idx_b];
        idx_b++;
        count++;

        while (sum > maxSum && idx_a > 0) {
            idx_a--;
            sum -= a[idx_a];
            count--;
        }

        if (sum <= maxSum && count > max_count) {
            max_count = count;
        }
    }

    return max_count;
}

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(fptr, "%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) {
```

```c
      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;
}
```

```c

    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;
    }
```

Output :


Input (stdin)

Download

```
1
5 4 10
4 2 4 6 1
2 1 8 5
```

Your Output (stdout)

```
4
```

Expected Output

Download

```
4
```