Nama : Arya Maulana

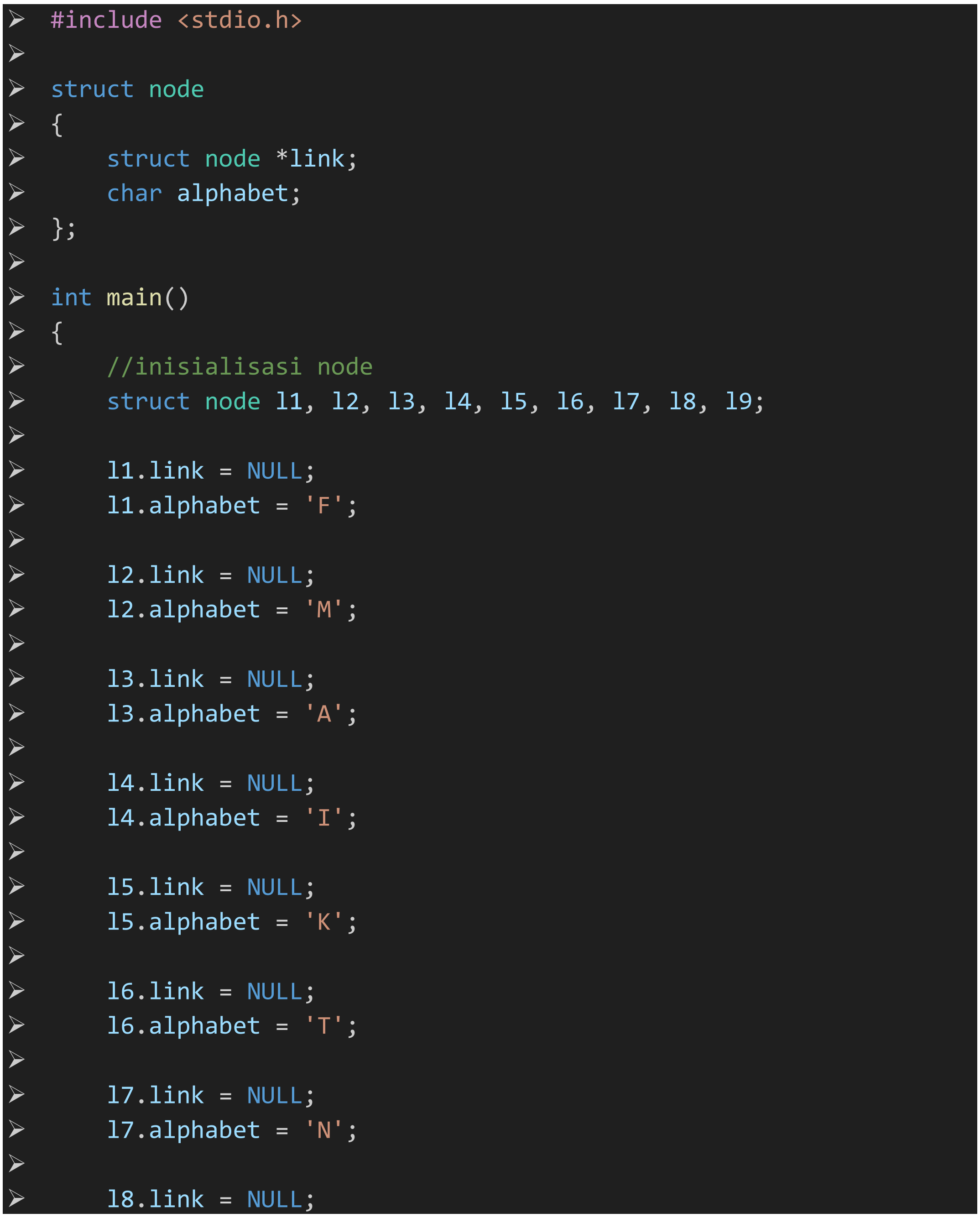
Kelas : IF 03-02

NIM : 1203230120

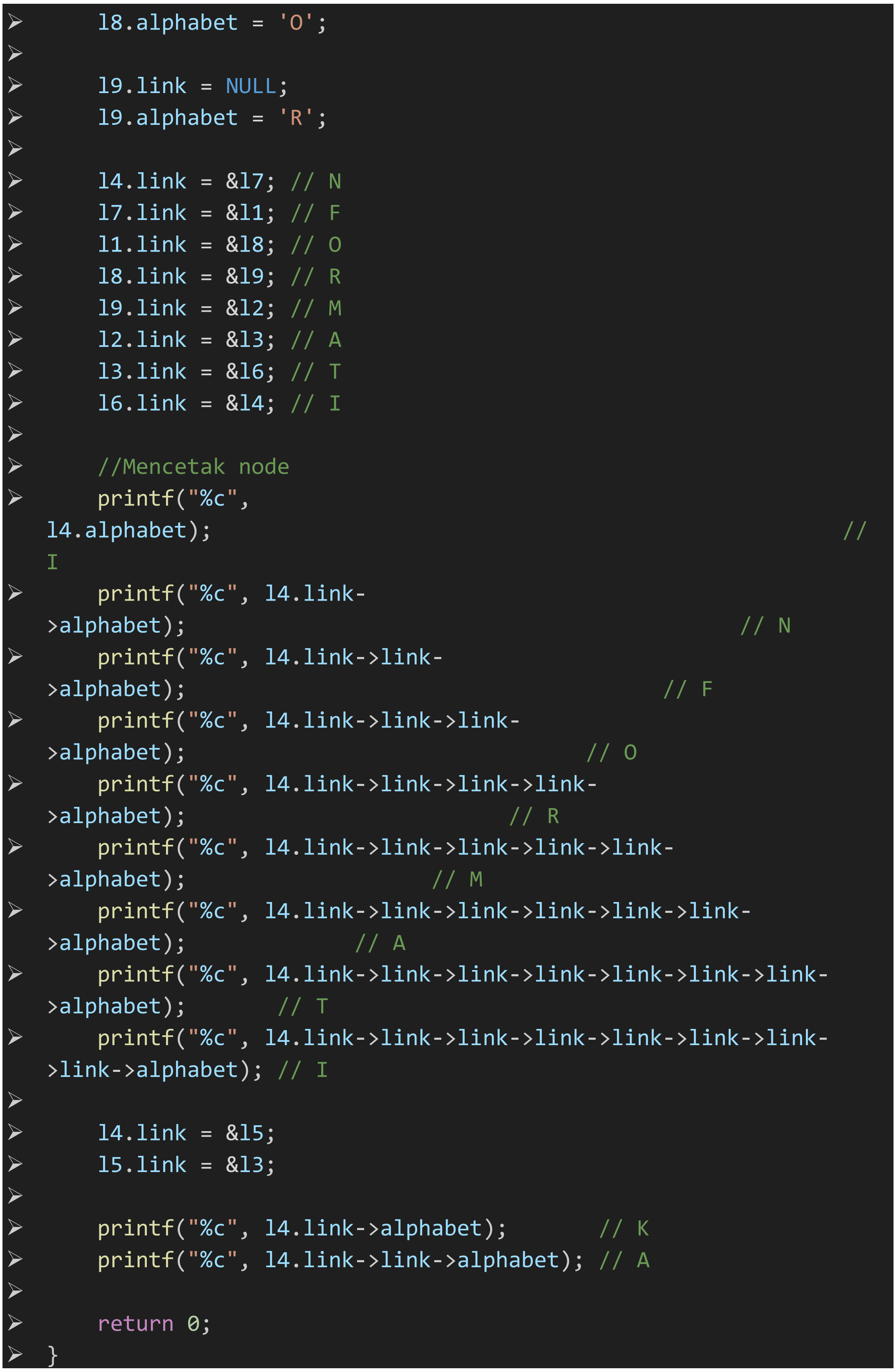
Tugas OTS Week 6

1) Soal Asisten Sherlock Holmes

- Source Code

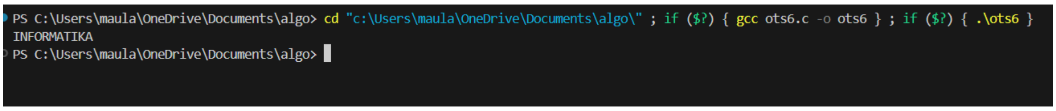


```
#include <stdio.h>

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

int main()
{
    //inisialisasi node
    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';
➢
➢      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
➢
➢      //Mencetak node
➢      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
➢
➢      return 0;
➢   }
```

- Hasil



```
PS C:\Users\maula\OneDrive\Documents\algo> cd "c:\Users\maula\OneDrive\Documents\algo\" ; if ($?) { gcc ots6.c -o ots6 } ; if ($?) { .\ots6 }
INFORMATIKA
PS C:\Users\maula\OneDrive\Documents\algo> 
```

- Penjelasan

  Pada codingan diatas, struct node memiliki variabel link sebagai pointer yaitu penghubung atau melanjutkan node berikutnya dan alphabet sebagai karakter pada node. Fungsi main mencetak node secara berurutan.

2) Soal Hackerrank

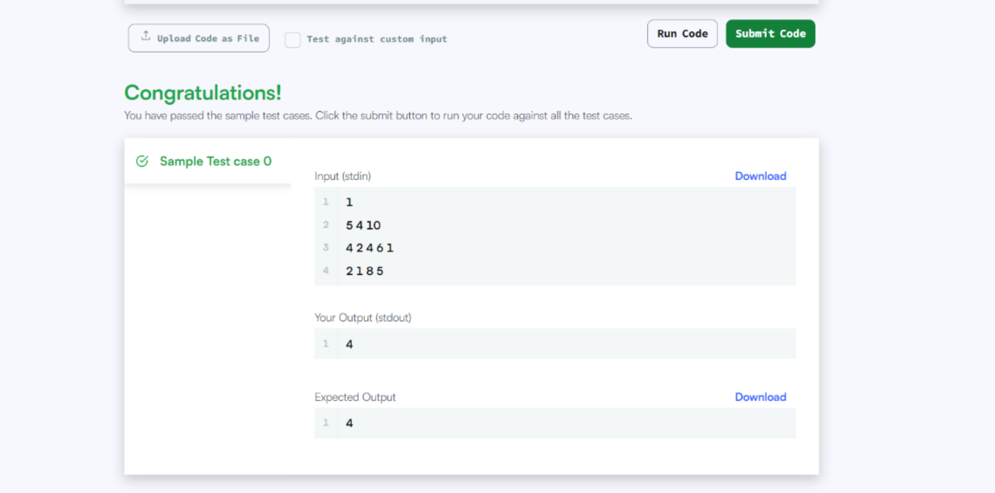
- Source Code

```
#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*);
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 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;
}

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);
    }

    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 = '\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;
}
```

- Hasil



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

Your Output (stdout)

```
4
```

Expected Output

```
4
```

- Penjelasan

  Codingan di atas untuk menentukan jumlah maksimum data yang dapat diambil dari dua tumpukan stack dengan batasan tidak melebihi nilai tertentu.