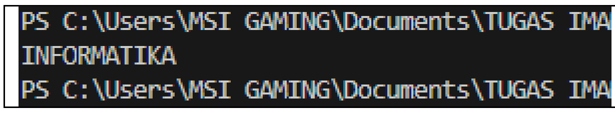
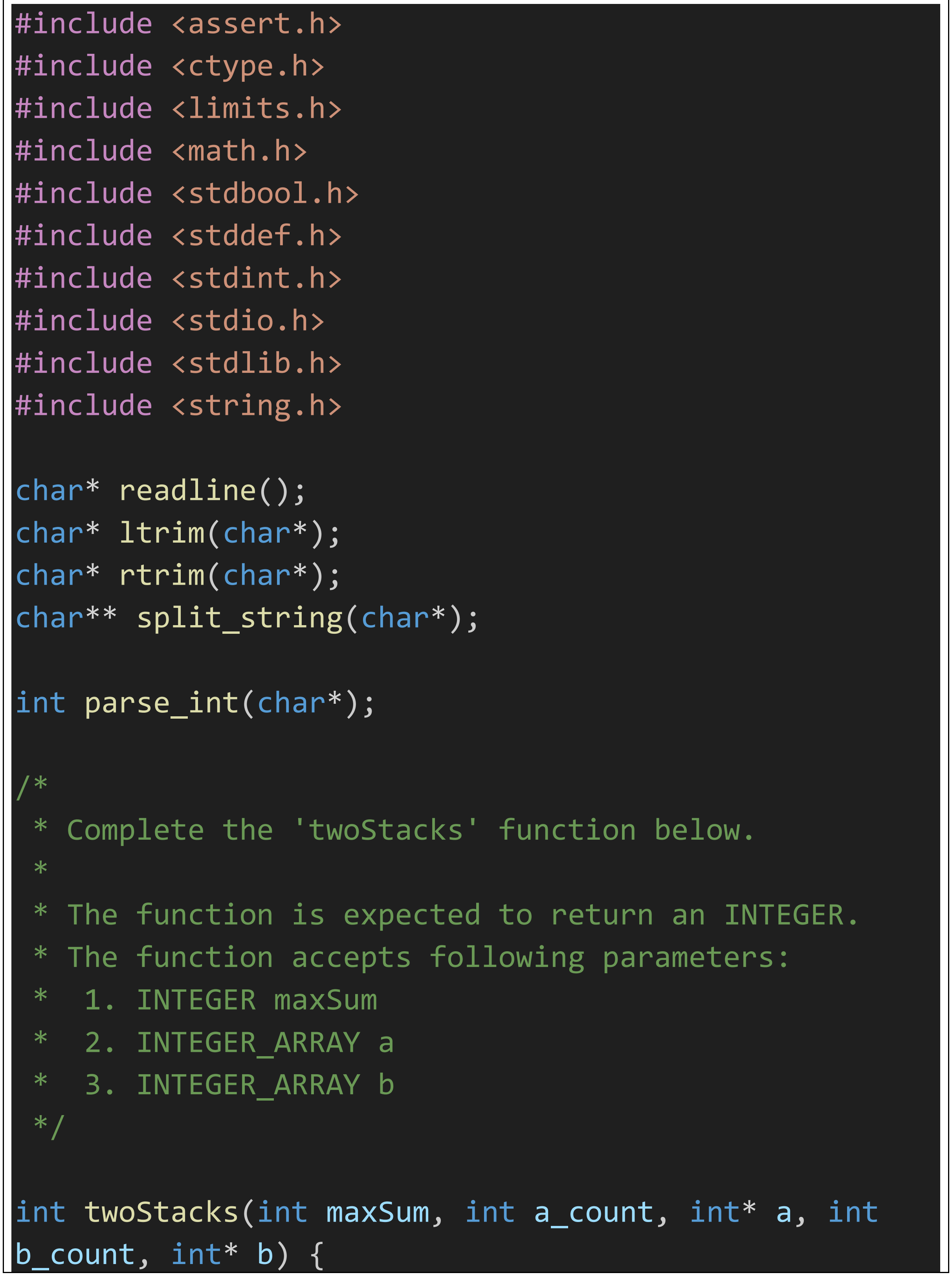
1. Asisten Sherlock Holmes

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

// Definisikan struktur batu
struct Stone {
    struct Stone* link;
    char* alphabet;
};

int main() {
    // Inisialisasi batu-batu
    struct Stone l1, l2, l3, l4, l5, l6, l7, l8, l9;

    // Inisialisasi huruf pada masing-masing batu
    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";
    // Hubungkan batu-batu sesuai arah panah
    l3.link = &l6;
    l6.link = &l9;
    l9.link = &l4;
    l4.link = &l7;
    l7.link = &l1;
    l1.link = &l8;
    l8.link = &l2;
    l2.link = &l5;
    l5.link = &l3;

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

    printf("\n");
}
```

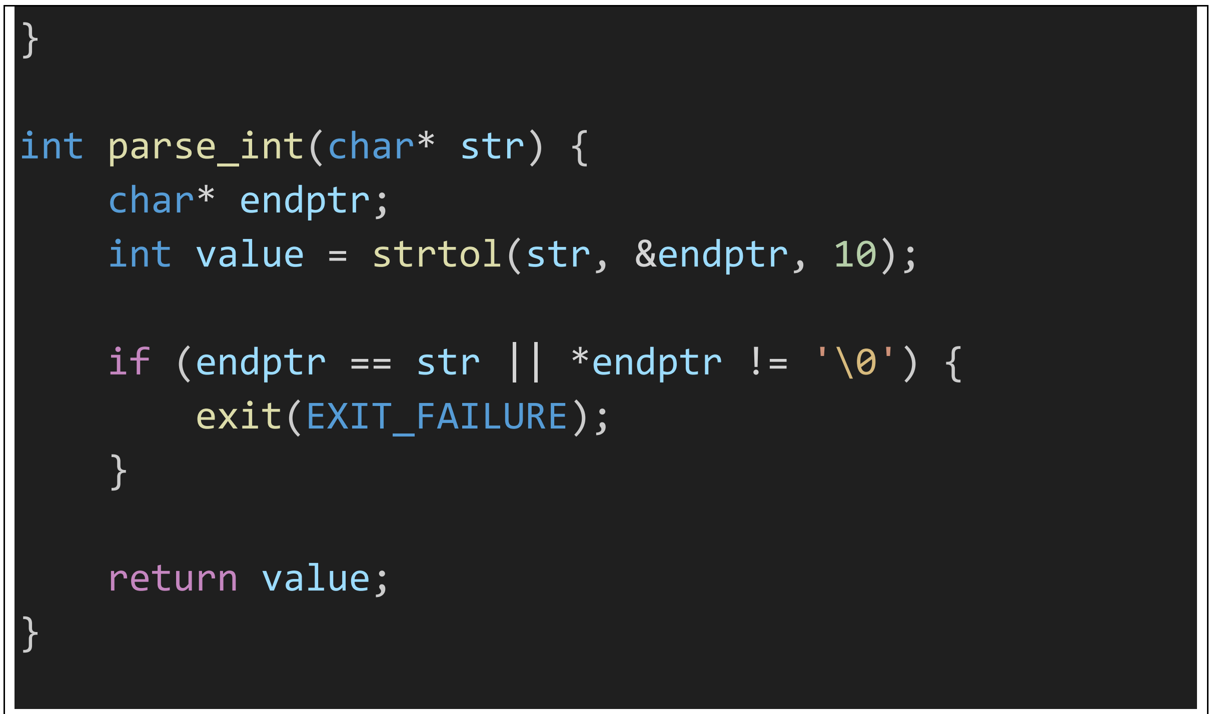
Output:


```
PS C:\Users\MSI GAMING\Documents\TUGAS IMA
INFORMATIKA
PS C:\Users\MSI GAMING\Documents\TUGAS IMA
```

2. HackerRank

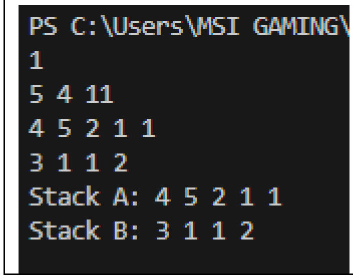

```c
#include <assert.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char* ltrim(char*);
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 sum = 0, count = 0;

    // Visualisasi stack a
    printf("Stack A: ");
    for (int i = 0; i < a_count; i++)
        printf("%d ", a[i]);
    printf("\n");

    // Visualisasi stack b
    printf("Stack B: ");
    for (int i = 0; i < b_count; i++)
        printf("%d ", b[i]);
    printf("\n\n");

    // Algoritma penyelesaian
    while (a_index < a_count && sum + a[a_index] <=
maxSum) {
        sum += a[a_index++];
        count++;
    }

    int max_count = count;

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

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

    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) {
```

```c
        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;
```

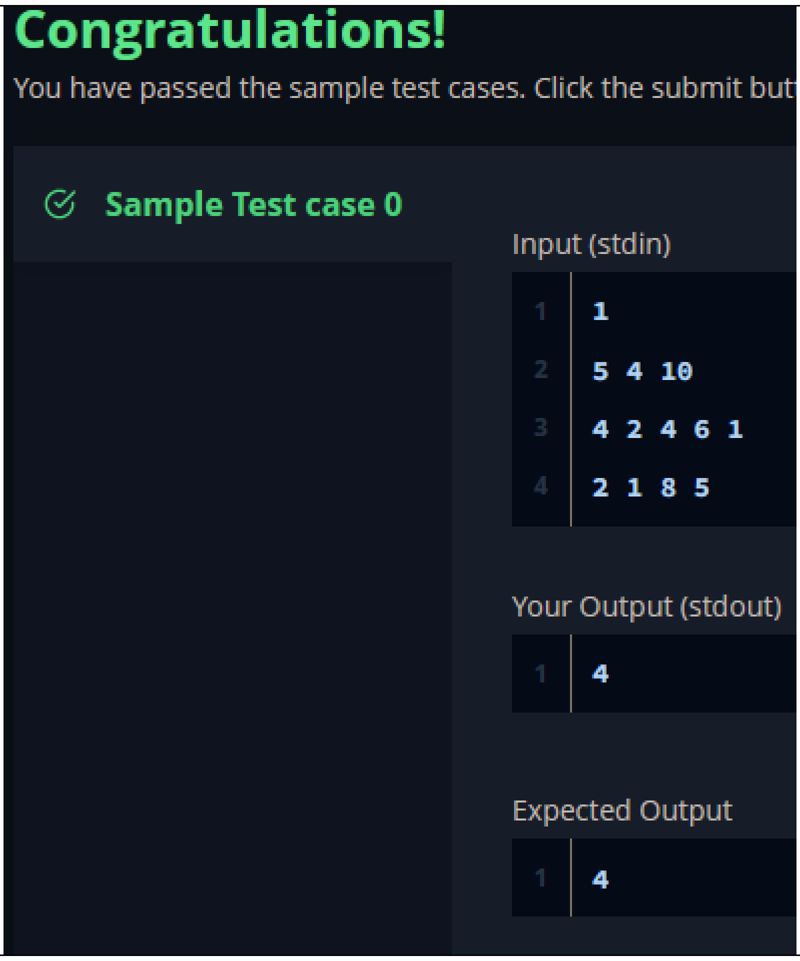


```c
}

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

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

    return value;
}
```

Output:



```
PS C:\Users\MSI GAMING\
1
5 4 11
4 5 2 1 1
3 1 1 2
Stack A: 4 5 2 1 1
Stack B: 3 1 1 2
```



Congratulations!

You have passed the sample test cases. Click the submit but

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
```