

Nama : Ahmad Wahyudi

NIM : 1203230116

Kelas : IF 03-02

Tugas algoritma dan struktur data

Source Code

```
#include <stdio.h>

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

int main()
{
    // Node initialization
    struct node 11, 12, 13, 14, 15, 16, 17, 18, 19;

    11.link = NULL;
    11.alphabet = 'F';

    12.link = NULL;
    12.alphabet = 'M';

    13.link = NULL;
    13.alphabet = 'A';
```

```
14.link = NULL;
```

```
14.alphabet = 'T';
```

```
15.link = NULL;
```

```
15.alphabet = 'K';
```

```
16.link = NULL;
```

```
16.alphabet = 'T';
```

```
17.link = NULL;
```

```
17.alphabet = 'N';
```

```
18.link = NULL;
```

```
18.alphabet = 'O';
```

```
19.link = NULL;
```

```
19.alphabet = 'R';
```

```
// Linking nodes
```

```
14.link = &17; // N
```

```
17.link = &11; // F
```

```
11.link = &18; // O
```

```
18.link = &19; // R
```

```
19.link = &l2; // M
```

```
12.link = &l3; // A
```

```
13.link = &l6; // T
```

```
16.link = &l4; // I
```

```
// Print linked list
```

```
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->link->alphabet); // A
```

```
printf("%c", l4.link->link->link->link->link->link->link->link->alphabet); // T
```

```
printf("%c", l4.link->link->link->link->link->link->link->link->link->alphabet); // I
```

```
14.link = &l5;
```

```
15.link = &l3;
```

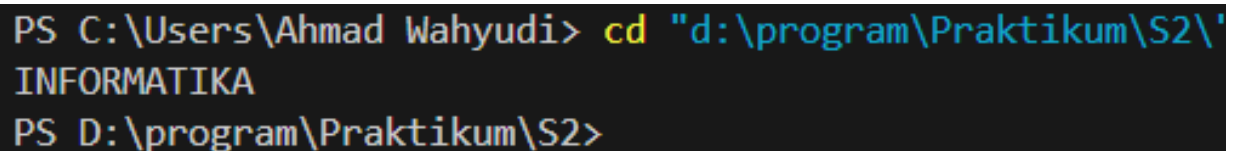
```
printf("%c", l4.link->alphabet); // K
```

```
printf("%c", l4.link->link->alphabet); // A

return 0;

}
```

Output



```
PS C:\Users\Ahmad Wahyudi> cd "d:\program\Praktikum\S2\INFORMATIKA"
PS D:\program\Praktikum\S2>
```

Penjelasan

Program ini mengimplementasikan linked list dalam program bahasa C.

- Include <stdio.h> berfungsi menjalankan input dan output.
- Struct node mempresentasikan elemen dalam linked list. Program ini terdiri dari 2 bagian :
 1. Link sebagai pointer.
 2. Alphabet sebagai variable.
- Main () berfungsi mendeklarasikan I1 sampai I9 yang masing-masing memiliki Link yang diatur sebagai Null.
- Linking noder berfungsi untuk mencetak linked list secara berurutan sesuai isi program.
- Print linked list berfungsi mencetak hasil dari program.

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 *getline();
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(fp, "%d\n", result);
}

fclose(fp);

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;
}

```

Output

Vs code

```
PS D:\program\Praktikum\S2> cd "d:\program\Praktikum\S2\"
1
5 4 11
4 5 2 1 1
3 1 1 2
PS D:\program\Praktikum\S2>
```

Hackerrank

The screenshot shows the Hackerrank interface after solving a challenge. At the top, a green banner says "Congratulations" and "You solved this challenge. Would you like to challenge your friends?". Below this, a list of test cases (0 to 6) is shown, all marked as successful. The "Compiler Message" section displays "Success". The "Input (stdin)" section shows four lines of input: "1", "5 4 10", "4 2 4 6 1", and "2 1 8 5". The "Expected Output" section shows the output "4".

Program ini memiliki fungsi twostacks berfungsi untuk menerima beberapa parameter sebagai input dan diharapkan mengembalikan nilai integer

1. Program menggunakan beberapa header file standar seperti `<assert.h>`, `<ctype.h>`, `<limits.h>`, `<math.h>`, `<stdbool.h>`, `<stddef.h>`, `<stdint.h>`, `<stdio.h>`, `<stdlib.h>`, dan `<string.h>`.
2. Program memiliki beberapa fungsi bantu seperti **getline**, **ltrim**, **rtrim**, **split_string**, dan **parse_int** untuk membantu dalam pengolahan input.

3. Fungsi utama dari program adalah **twoStacks**, yang menghitung jumlah elemen dari dua tumpukan (**a** dan **b**) sedemikian sehingga jumlah elemen yang diambil dari kedua tumpukan tidak melebihi **maxSum**.
4. Program juga memiliki fungsi **main** yang digunakan untuk membaca input dari pengguna dan menjalankan fungsi **twoStacks** untuk menghitung hasilnya.
5. Program menghasilkan output dalam format yang sesuai dengan spesifikasi yang diberikan.