

Agastya Ferdian Putra Rahardjo

IF-03-01/1203230124

Latihan Soal OTH

1. Source Code :

```
2. #include <stdio.h>
3. #include <stdlib.h>
4.
5. struct Node {
6.     char* alphabet;
7.     struct Node* link;
8. };
9.
10. int main() {
11.     // Deklarasi node-node
12.     struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
13.     struct Node *link, *l3ptr;
14.
15.     // Inisialisasi node-node dengan menggunakan potongan kode soal
16.     l1.link = NULL;
17.     l1.alphabet = "F";
18.
19.     l2.link = NULL;
20.     l2.alphabet = "M";
21.
22.     l3.link = NULL;
23.     l3.alphabet = "A";
24.
25.     l4.link = NULL;
26.     l4.alphabet = "I";
27.
28.     l5.link = NULL;
29.     l5.alphabet = "K";
30.
31.     l6.link = NULL;
32.     l6.alphabet = "T";
33.
34.     l7.link = NULL;
35.     l7.alphabet = "N";
36.
37.     l8.link = NULL;
38.     l8.alphabet = "O";
39.
40.     l9.link = NULL;
41.     l9.alphabet = "R";
42.
```

```

43. // Mengatur koneksi antar node sesuai dengan urutan yang diinginkan
44. l7.link = &l1; // Menyambungkan ke l1
45. l1.link = &l8; // Menyambungkan ke l1
46. l8.link = &l2; // Menyambungkan ke l1
47. l2.link = &l5; // Menyambungkan ke l1
48. l5.link = &l3; // Menyambungkan ke l1
49. l3.link = &l6; // Menyambungkan ke l1
50. l6.link = &l9;
51. l9.link = &l4;
52. l4.link = &l7;
53.
54. // Starting point
55. l3ptr = &l7;
56.
57. // Akses data menggunakan printf
58. printf("%s", l3.link->link->link->alphabet); // Menampilkan huruf I
59. printf("%s", l3.link->link->link->link->alphabet); // Menampilkan
    huruf N
60. printf("%s", l3.link->link->link->link->link->alphabet); //
    Menampilkan huruf F
61. printf("%s", l3.link->link->link->link->link->link->alphabet); //
    Menampilkan huruf O
62. printf("%s", l3.link->link->alphabet); // Menampilkan huruf R
63. printf("%s", l3.link->link->link->link->link->link->link->
    >alphabet); // Menampilkan huruf M
64. printf("%s", l3.alphabet); // Menampilkan huruf A
65. printf("%s", l3.link->alphabet); // Menampilkan huruf T
66. printf("%s", l3.link->link->link->alphabet); // Menampilkan huruf I
67. printf("%s", l3.link->link->link->link->link->link->link->
    >alphabet); // Menampilkan huruf K
68. printf("%s", l3.alphabet); // Menampilkan huruf A
69.
70. return 0;
71. }

```

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
ASPRATI
PS C:\Users\Agastya\Documents\File AgastyaGtg>

```

2. Source Code :

```
3. #include <stdio.h>
4. #include <stdlib.h>
5. #include <stdbool.h>
6. #include <string.h>
7.
8. char* readline();
9. char* ltrim(char*);
10. char* rtrim(char*);
11. char** split_string(char*);
12. int parse_int(char*);
13.
14. int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
15.     int count = 0;
16.     int sum = 0;
17.     int idx_a = 0, idx_b = 0;
18.
19.     while (idx_a < a_count && sum + a[idx_a] <= maxSum) {
20.         sum += a[idx_a];
21.         idx_a++;
22.         count++;
23.     }
24.
25.     int max_count = count;
26.
27.     while (idx_b < b_count && idx_a >= 0) {
28.         sum += b[idx_b];
29.         idx_b++;
30.         count++;
31.
32.         while (sum > maxSum && idx_a > 0) {
33.             idx_a--;
34.             sum -= a[idx_a];
35.             count--;
36.         }
37.
38.         if (sum <= maxSum && count > max_count) {
39.             max_count = count;
40.         }
41.     }
42.
43.     return max_count;
44. }
45.
46. int main()
47. {
48.     FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");
49. }
```

```

50.     int g = parse_int(ltrim(rtrim(readline())));
51.
52.     for (int g_itr = 0; g_itr < g; g_itr++) {
53.         char** first_multiple_input = split_string(rtrim(readline()));
54.
55.         int n = parse_int(*(first_multiple_input + 0));
56.
57.         int m = parse_int(*(first_multiple_input + 1));
58.
59.         int maxSum = parse_int(*(first_multiple_input + 2));
60.
61.         char** a_temp = split_string(rtrim(readline()));
62.         int* a = malloc(n * sizeof(int));
63.         for (int i = 0; i < n; i++) {
64.             int a_item = parse_int(*(a_temp + i));
65.             *(a + i) = a_item;
66.         }
67.
68.         char** b_temp = split_string(rtrim(readline()));
69.         int* b = malloc(m * sizeof(int));
70.         for (int i = 0; i < m; i++) {
71.             int b_item = parse_int(*(b_temp + i));
72.             *(b + i) = b_item;
73.         }
74.
75.         int result = twoStacks(maxSum, n, a, m, b);
76.
77.         fprintf(fp_ptr, "%d\n", result);
78.
79.         free(a);
80.         free(b);
81.     }
82.
83.     fclose(fp_ptr);
84.
85.     return 0;
86. }
87.
88. char* readline() {
89.     size_t alloc_length = 1024;
90.     size_t data_length = 0;
91.     char* data = malloc(alloc_length);
92.
93.     while (true) {
94.         char* cursor = data + data_length;
95.         char* line = fgets(cursor, alloc_length - data_length, stdin);
96.
97.         if (!line) {

```

```

98.         break;
99.     }
100.
101.         data_length += strlen(cursor);
102.
103.         if (data_length < alloc_length - 1 || data[data_length -
1] == '\n') {
104.             break;
105.         }
106.
107.         alloc_length <= 1;
108.         data = realloc(data, alloc_length);
109.
110.         if (!data) {
111.             data = '\0';
112.             break;
113.         }
114.     }
115.
116.     if (data[data_length - 1] == '\n') {
117.         data[data_length - 1] = '\0';
118.         data = realloc(data, data_length);
119.         if (!data) {
120.             data = '\0';
121.         }
122.     } else {
123.         data = realloc(data, data_length + 1);
124.         if (!data) {
125.             data = '\0';
126.         } else {
127.             data[data_length] = '\0';
128.         }
129.     }
130.     return data;
131. }
132.
133. char* ltrim(char* str) {
134.     if (!str) {
135.         return '\0';
136.     }
137.     if (!*str) {
138.         return str;
139.     }
140.     while (*str != '\0' && isspace(*str)) {
141.         str++;
142.     }
143.     return str;
144. }

```

```
145.
146.     char* rtrim(char* str) {
147.         if (!str) {
148.             return '\0';
149.         }
150.         if (!*str) {
151.             return str;
152.         }
153.         char* end = str + strlen(str) - 1;
154.         while (end >= str && isspace(*end)) {
155.             end--;
156.         }
157.         *(end + 1) = '\0';
158.         return str;
159.     }
160.
161.     char** split_string(char* str) {
162.         char** splits = NULL;
163.         char* token = strtok(str, " ");
164.         int spaces = 0;
165.         while (token) {
166.             splits = realloc(splits, sizeof(char*) * ++spaces);
167.             if (!splits) {
168.                 return splits;
169.             }
170.             splits[spaces - 1] = token;
171.             token = strtok(NULL, " ");
172.         }
173.         return splits;
174.     }
175.
176.     int parse_int(char* str) {
177.         char* endptr;
178.         int value = strtol(str, &endptr, 10);
179.         if (endptr == str || *endptr != '\0') {
180.             exit(EXIT_FAILURE);
181.         }
182.         return value;
183.     }
```

Output :

Input (stdin)

[Download](#)

1	1
2	5 4 10
3	4 2 4 6 1
4	2 1 8 5

Your Output (stdout)

1	4
---	---

Expected Output

[Download](#)

1	4
---	---