

Nama: Jonathan Ferdinand Mayon

Tugas OTH Struct dan Stack

Kelas: IF 03 03

Nim: 1203230087

Tugas 1

1. Source Code

```
2. #include <stdio.h>
3.
4. struct node
5. {
6.     struct node *link;
7.     char alphabet;
8. };
9.
10. int main()
11. {
12.     // Node initialization
13.     struct node l1, l2, l3, l4, l5, l6, l7, l8, l9;
14.
15.     l1.link = NULL;
16.     l1.alphabet = 'F';
17.
18.     l2.link = NULL;
19.     l2.alphabet = 'M';
20.
21.     l3.link = NULL;
22.     l3.alphabet = 'A';
23.
24.     l4.link = NULL;
25.     l4.alphabet = 'I';
26.
27.     l5.link = NULL;
28.     l5.alphabet = 'K';
29.
30.     l6.link = NULL;
31.     l6.alphabet = 'T';
32.
33.     l7.link = NULL;
34.     l7.alphabet = 'N';
35.
36.     l8.link = NULL;
37.     l8.alphabet = 'O';
```

```

38.
39.     19.link = NULL;
40.     19.alphabet = 'R';
41.
42.     // Linking nodes
43.     14.link = &17; // N
44.     17.link = &11; // F
45.     11.link = &18; // O
46.     18.link = &19; // R
47.     19.link = &12; // M
48.     12.link = &13; // A
49.     13.link = &16; // T
50.     16.link = &14; // I
51.
52.     // Print linked list
53.     printf("%c",
14.alphabet); // I
54.     printf("%c", 14.link-
>alphabet); // N
55.     printf("%c", 14.link->link-
>alphabet); // F
56.     printf("%c", 14.link->link->link-
>alphabet); // O
57.     printf("%c", 14.link->link->link->link-
>alphabet); // R
58.     printf("%c", 14.link->link->link->link->link-
>alphabet); // M
59.     printf("%c", 14.link->link->link->link->link->link-
>alphabet); // A
60.     printf("%c", 14.link->link->link->link->link->link->link-
>alphabet); // T
61.     printf("%c", 14.link->link->link->link->link->link->link->link-
>alphabet); // I
62.
63.     14.link = &15;
64.     15.link = &13;
65.
66.     printf("%c", 14.link->alphabet); // K
67.     printf("%c", 14.link->link->alphabet); // A
68.
69.     return 0;
70. }

```

2. Output

```
PS C:\project> cd "c:\project\Kuliah Tel-  
U\kuliah sem 2\ASD PRAKTIKUM\" ; if ($?)  
{ gcc tempCodeRunnerFile.c -o tempCodeRun  
nerFile } ; if ($?) { .\tempCodeRunnerFil  
e }  
INFORMATIKA  
PS C:\project\Kuliah Tel-U\kuliah sem 2\A  
SD PRAKTIKUM>
```

Penjelas code:

1. `#include <stdio.h>`: Ini adalah direktif preprocessor yang memasukkan konten dari file header `stdio.h` ke dalam program. File header ini berisi deklarasi fungsi standar input-output seperti `printf()` dan `scanf()`.
2. `struct node { struct node *link; char alphabet; };`: Ini mendefinisikan sebuah struktur data bernama `node`, yang memiliki dua anggota. `link` adalah pointer ke node berikutnya dalam linked list, dan `alphabet` menyimpan satu karakter.
3. `int main() { ... }`: Ini adalah fungsi utama dari program.
4. `struct node l1, l2, l3, l4, l5, l6, l7, l8, l9;`: Ini mendeklarasikan sembilan variabel bertipe `struct node` yang akan digunakan sebagai node-node dalam linked list.
5. Inisialisasi setiap node dengan menetapkan `link` ke `NULL` dan karakter ke nilai tertentu.
6. Menghubungkan node-node untuk membentuk linked list.
7. Mencetak isi linked list dengan mencetak karakter dari setiap node berurutan, dimulai dari
8. Mengubah hubungan antar node dengan menetapkan pointer `link` dari beberapa node yang dipilih ke node-node lainnya.
9. Mencetak lagi isi linked list setelah perubahan.
10. `return 0;`: Mengembalikan nilai 0 yang menunjukkan bahwa program telah berakhir dengan sukses. Ini adalah akhir dari fungsi `main()` dan akhir dari program.

Tugas 2

1. Source code

```
2. #include <assert.h>
3. #include <ctype.h>
4. #include <limits.h>
5. #include <math.h>
6. #include <stdbool.h>
7. #include <stddef.h>
8. #include <stdint.h>
9. #include <stdio.h>
10. #include <stdlib.h>
11. #include <string.h>
12.
13. char *getline();
14. char *ltrim(char *);
15. char *rtrim(char *);
16. char **split_string(char *);
17.
18. int parse_int(char *);
19.
20. /*
21.  * Complete the 'twoStacks' function below.
22.  *
23.  * The function is expected to return an INTEGER.
24.  * The function accepts following parameters:
25.  * 1. INTEGER maxSum
26.  * 2. INTEGER_ARRAY a
27.  * 3. INTEGER_ARRAY b
28.  */
29.
30. int twoStacks(int maxSum, int a_count, int *a, int b_count, int *b)
31. {
32.     int i = 0, j = 0, sum = 0, count = 0;
33.     while (i < a_count && sum + a[i] <= maxSum)
34.     {
35.         sum += a[i];
36.         i++;
37.     }
38.     count = i;
39.     while (j < b_count && i >= 0)
40.     {
41.         sum += b[j];
42.         j++;
43.         while (sum > maxSum && i > 0)
```

```
44.     {
45.         i--;
46.         sum -= a[i];
47.     }
48.     if (sum <= maxSum && i + j > count)
49.     {
50.         count = i + j;
51.     }
52. }
53. return count;
54.}
55.
56.int main()
57.{
58.    FILE *fptr = fopen(getenv("OUTPUT_PATH"), "w");
59.
60.    int g = parse_int(ltrim(rtrim(readline())));
61.
62.    for (int g_itr = 0; g_itr < g; g_itr++)
63.    {
64.        char **first_multiple_input = split_string(rtrim(readline()));
65.
66.        int n = parse_int(*(first_multiple_input + 0));
67.
68.        int m = parse_int(*(first_multiple_input + 1));
69.
70.        int maxSum = parse_int(*(first_multiple_input + 2));
71.
72.        char **a_temp = split_string(rtrim(readline()));
73.
74.        int *a = malloc(n * sizeof(int));
75.
76.        for (int i = 0; i < n; i++)
77.        {
78.            int a_item = parse_int(*(a_temp + i));
79.
80.            *(a + i) = a_item;
81.        }
82.
83.        char **b_temp = split_string(rtrim(readline()));
84.
85.        int *b = malloc(m * sizeof(int));
86.
87.        for (int i = 0; i < m; i++)
88.        {
```

```

89.         int b_item = parse_int(*(b_temp + i));
90.
91.         *(b + i) = b_item;
92.     }
93.
94.     int result = twoStacks(maxSum, n, a, m, b);
95.
96.     fprintf(fp_ptr, "%d\n", result);
97. }
98.
99. fclose(fp_ptr);
100.
101.     return 0;
102. }
103.
104. char *readline()
105. {
106.     size_t alloc_length = 1024;
107.     size_t data_length = 0;
108.
109.     char *data = malloc(alloc_length);
110.
111.     while (true)
112.     {
113.         char *cursor = data + data_length;
114.         char *line = fgets(cursor, alloc_length - data_length,
115.             stdin);
116.
117.         if (!line)
118.         {
119.             break;
120.         }
121.
122.         data_length += strlen(cursor);
123.
124.         if (data_length < alloc_length - 1 || data[data_length - 1]
125.             == '\n')
126.         {
127.             break;
128.         }
129.
130.         alloc_length <= 1;
131.
132.         data = realloc(data, alloc_length);

```

```
132.         if (!data)
133.         {
134.             data = '\0';
135.
136.             break;
137.         }
138.     }
139.
140.     if (data[data_length - 1] == '\n')
141.     {
142.         data[data_length - 1] = '\0';
143.
144.         data = realloc(data, data_length);
145.
146.         if (!data)
147.         {
148.             data = '\0';
149.         }
150.     }
151.     else
152.     {
153.         data = realloc(data, data_length + 1);
154.
155.         if (!data)
156.         {
157.             data = '\0';
158.         }
159.         else
160.         {
161.             data[data_length] = '\0';
162.         }
163.     }
164.
165.     return data;
166. }
167.
168. char *ltrim(char *str)
169. {
170.     if (!str)
171.     {
172.         return '\0';
173.     }
174.
175.     if (!*str)
176.     {
```

```
177.         return str;
178.     }
179.
180.     while (*str != '\0' && isspace(*str))
181.     {
182.         str++;
183.     }
184.
185.     return str;
186. }
187.
188. char *rtrim(char *str)
189. {
190.     if (!str)
191.     {
192.         return '\0';
193.     }
194.
195.     if (!*str)
196.     {
197.         return str;
198.     }
199.
200.     char *end = str + strlen(str) - 1;
201.
202.     while (end >= str && isspace(*end))
203.     {
204.         end--;
205.     }
206.
207.     *(end + 1) = '\0';
208.
209.     return str;
210. }
211.
212. char **split_string(char *str)
213. {
214.     char **splits = NULL;
215.     char *token = strtok(str, " ");
216.
217.     int spaces = 0;
218.
219.     while (token)
220.     {
221.         splits = realloc(splits, sizeof(char *) * ++spaces);
```




```

222.
223.         if (!splits)
224.         {
225.             return splits;
226.         }
227.
228.         splits[spaces - 1] = token;
229.
230.         token = strtok(NULL, " ");
231.     }
232.
233.     return splits;
234. }
235.
236. int parse_int(char *str)
237. {
238.     char *endptr;
239.     int value = strtol(str, &endptr, 10);
240.
241.     if (endptr == str || *endptr != '\0')
242.     {
243.         exit(EXIT_FAILURE);
244.     }
245.
246.     return value;
247. }

```

2. Ss Hacker Rank



You have earned 30.00 points!
You are now 70 points away from the 2nd star for your problem solving badge.

0% 30/100

Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[Next Challenge](#)

- Test case 0
- Test case 1
- Test case 2
- Test case 3
- Test case 4
- Test case 5
- Test case 6

Compiler Message

Success

Input (stdin) [Download](#)

```

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

```

Expected Output [Download](#)

```

1 4

```

Penjelas code:

1. Fungsi `twoStacks` dimulai dengan inisialisasi beberapa variabel, termasuk `i` dan `j` untuk mengindeks tumpukan `a` dan `b` secara terpisah, serta variabel `sum` untuk melacak jumlah elemen yang telah diambil dari tumpukan.
2. Selanjutnya, program menggunakan loop `while` untuk mengambil elemen dari tumpukan `a` sebanyak mungkin hingga jumlahnya melebihi atau sama dengan `maxSum`. Ini dilakukan dengan menambahkan setiap elemen dari tumpukan `a` ke `sum`.
3. Program kemudian menggunakan loop `while` lainnya untuk menambahkan elemen dari tumpukan `b` ke `sum`, sambil memperbarui `i` (indeks tumpukan `a`) jika jumlahnya melebihi `maxSum`. Hal ini dilakukan untuk menemukan kombinasi maksimum dari elemen-elemen dari kedua tumpukan yang memenuhi batasan `maxSum`.
4. Setelah menemukan jumlah maksimum yang memenuhi batasan `maxSum`, program mengembalikan nilai tersebut.
5. Fungsi `main` digunakan untuk membaca input dari stdin, memanggil fungsi `twoStacks` untuk setiap kasus uji, dan menulis hasilnya ke stdout.
6. Fungsi bantuan seperti `getline`, `ltrim`, `rtrim`, `split_string`, dan `parse_int` digunakan untuk membantu dalam pengolahan input dan output.
7. Terakhir, program ditutup dengan menutup file output yang telah dibuka dan mengembalikan nilai 0 untuk menandakan bahwa program telah berakhir dengan sukses.