

NIM : 1203230105

SS dan Penjelasan

```

1  #include <stdio.h>
2
3  // deklarasi struct
4  struct elemen {
5      char alphabet;
6      struct elemen *link; // pointer ke link sesuai urutan yang telah di atur di bawah
7  };
8
9  int main() {
10     // deklarasi variabel dari l1 - l9
11     struct elemen l1, l2, l3, l4, l5, l6, l7, l8, l9;
12     l1.link = NULL; // jika melakukan link ke var lain maka nilai var sebelumnya menjadi NULL
13     l1.alphabet = 'F'; // jika di assign ke alphabet maka cetak huruf yang sudah di tentukan
14     l2.link = NULL;
15     l2.alphabet = 'M';
16     l3.link = NULL;
17     l3.alphabet = 'A';
18     l4.link = NULL;
19     l4.alphabet = 'I';
20     l5.link = NULL;
21     l5.alphabet = 'K';
22     l6.link = NULL;
23     l6.alphabet = 'T';
24     l7.link = NULL;
25     l7.alphabet = 'N';
26     l8.link = NULL;
27     l8.alphabet = 'O';
28     l9.link = NULL;
29     l9.alphabet = 'R';
30
31     // melakukan link antar elemen yang dimulai dari l7 yang berupa huruf N sampai l4 yang berupa huruf I
32     l7.link = &l1;
33     l8.link = &l2;
34     l9.link = &l3;
35     l2.link = &l4;
36     l3.link = &l5;
37     l4.link = &l6;
38     l5.link = &l7;
39     l6.link = &l8;
40     l7.link = &l9;

```

The screenshot shows a Windows IDE with a dark theme. The Explorer pane on the left lists files: C2, .vscode, output, C1.c, C OTH 1,2.c, C OTH 1.c, C OTH 2,2.c, C OTH 2.c (selected), C OTS 1.c, C OTS 2.c, C OTS 3.c, C OTS 4.c, C P1.1.c, C P1.2.c, C P1.3.c, C P1.4.c, C Quiz2.c, stack.c, Struct.c, and Switch.c. The main editor displays the code for C OTH 2.c, which is a C program for a linked list encryption algorithm. The code defines a linked list structure with a head pointer and a list of nodes. It then performs a series of operations to encrypt the list, including reversing the list and applying a key-based encryption function. The program outputs the encrypted list and the key used for encryption.

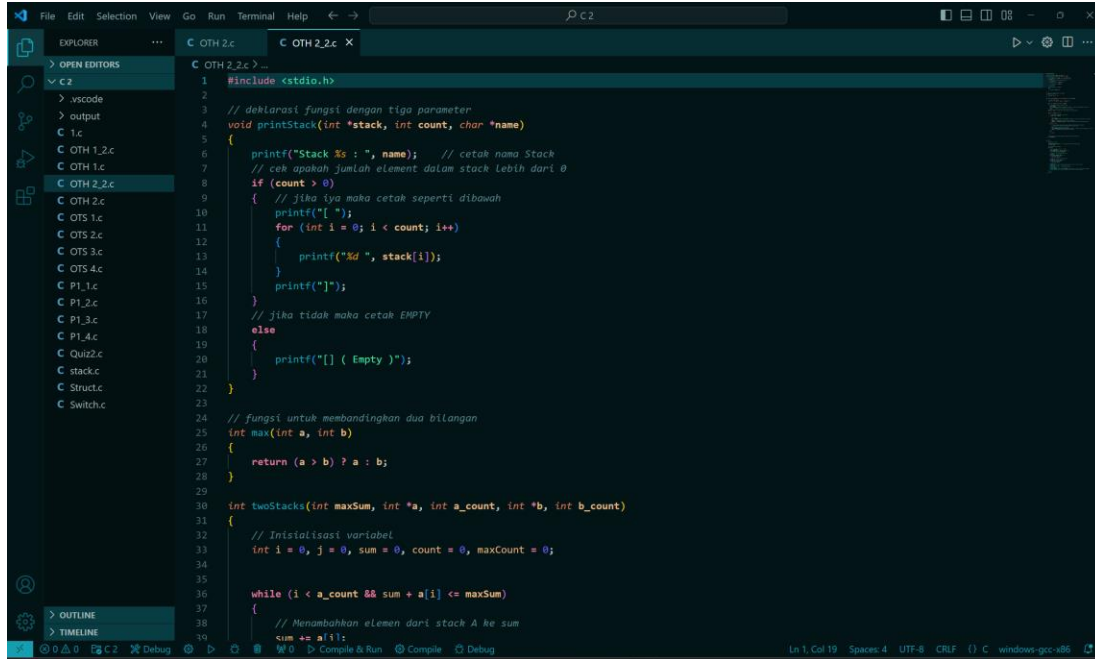
```
1 int main() {
2     // ...
3     // ...
4     // ...
5     // ...
6     // ...
7     // ...
8     // ...
9     // ...
10    // ...
11    // ...
12    // ...
13    // ...
14    // ...
15    // ...
16    // ...
17    // ...
18    // ...
19    // ...
20    // ...
21    // ...
22    // ...
23    // ...
24    // ...
25    // ...
26    // ...
27    // ...
28    // ...
29    // ...
30    // ...
31    // ...
32    // ...
33    // ...
34    // ...
35    // ...
36    // ...
37    // ...
38    // ...
39    // ...
40    // ...
41    // ...
42    // ...
43    // ...
44    // ...
45    // ...
46    // ...
47    // ...
48    // ...
49    // ...
50    // ...
51    // ...
52    // ...
53    // ...
54    // ...
55    // ...
56    // ...
57    // ...
58    // ...
59    // ...
60    // ...
61    // ...
62    // ...
63    // ...
64    // ...
65    // ...
66    // ...
67    // ...
68    // ...
69    // ...
70    // ...
71    // ...
72    // ...
73    // ...
74    // ...
75    // ...
76    // ...
77    // ...
78    // ...
79    // ...
80    // ...
81    // ...
82    // ...
83    // ...
84    // ...
85    // ...
86    // ...
87    // ...
88    // ...
89    // ...
90    // ...
91    // ...
92    // ...
93    // ...
94    // ...
95    // ...
96    // ...
97    // ...
98    // ...
99    // ...
100   // ...
101   // ...
102   // ...
103   // ...
104   // ...
105   // ...
106   // ...
107   // ...
108   // ...
109   // ...
110   // ...
111   // ...
112   // ...
113   // ...
114   // ...
115   // ...
116   // ...
117   // ...
118   // ...
119   // ...
120   // ...
121   // ...
122   // ...
123   // ...
124   // ...
125   // ...
126   // ...
127   // ...
128   // ...
129   // ...
130   // ...
131   // ...
132   // ...
133   // ...
134   // ...
135   // ...
136   // ...
137   // ...
138   // ...
139   // ...
140   // ...
141   // ...
142   // ...
143   // ...
144   // ...
145   // ...
146   // ...
147   // ...
148   // ...
149   // ...
150   // ...
151   // ...
152   // ...
153   // ...
154   // ...
155   // ...
156   // ...
157   // ...
158   // ...
159   // ...
160   // ...
161   // ...
162   // ...
163   // ...
164   // ...
165   // ...
166   // ...
167   // ...
168   // ...
169   // ...
170   // ...
171   // ...
172   // ...
173   // ...
174   // ...
175   // ...
176   // ...
177   // ...
178   // ...
179   // ...
180   // ...
181   // ...
182   // ...
183   // ...
184   // ...
185   // ...
186   // ...
187   // ...
188   // ...
189   // ...
190   // ...
191   // ...
192   // ...
193   // ...
194   // ...
195   // ...
196   // ...
197   // ...
198   // ...
199   // ...
200   // ...
201   // ...
202   // ...
203   // ...
204   // ...
205   // ...
206   // ...
207   // ...
208   // ...
209   // ...
210   // ...
211   // ...
212   // ...
213   // ...
214   // ...
215   // ...
216   // ...
217   // ...
218   // ...
219   // ...
220   // ...
221   // ...
222   // ...
223   // ...
224   // ...
225   // ...
226   // ...
227   // ...
228   // ...
229   // ...
230   // ...
231   // ...
232   // ...
233   // ...
234   // ...
235   // ...
236   // ...
237   // ...
238   // ...
239   // ...
240   // ...
241   // ...
242   // ...
243   // ...
244   // ...
245   // ...
246   // ...
247   // ...
248   // ...
249   // ...
250   // ...
251   // ...
252   // ...
253   // ...
254   // ...
255   // ...
256   // ...
257   // ...
258   // ...
259   // ...
260   // ...
261   // ...
262   // ...
263   // ...
264   // ...
265   // ...
266   // ...
267   // ...
268   // ...
269   // ...
270   // ...
271   // ...
272   // ...
273   // ...
274   // ...
275   // ...
276   // ...
277   // ...
278   // ...
279   // ...
280   // ...
281   // ...
282   // ...
283   // ...
284   // ...
285   // ...
286   // ...
287   // ...
288   // ...
289   // ...
290   // ...
291   // ...
292   // ...
293   // ...
294   // ...
295   // ...
296   // ...
297   // ...
298   // ...
299   // ...
300   // ...
301   // ...
302   // ...
303   // ...
304   // ...
305   // ...
306   // ...
307   // ...
308   // ...
309   // ...
310   // ...
311   // ...
312   // ...
313   // ...
314   // ...
315   // ...
316   // ...
317   // ...
318   // ...
319   // ...
320   // ...
321   // ...
322   // ...
323   // ...
324   // ...
325   // ...
326   // ...
327   // ...
328   // ...
329   // ...
330   // ...
331   // ...
332   // ...
333   // ...
334   // ...
335   // ...
336   // ...
337   // ...
338   // ...
339   // ...
340   // ...
341   // ...
342   // ...
343   // ...
344   // ...
345   // ...
346   // ...
347   // ...
348   // ...
349   // ...
350   // ...
351   // ...
352   // ...
353   // ...
354   // ...
355   // ...
356   // ...
357   // ...
358   // ...
359   // ...
360   // ...
361   // ...
362   // ...
363   // ...
364   // ...
365   // ...
366   // ...
367   // ...
368   // ...
369   // ...
370   // ...
371   // ...
372   // ...
373   // ...
374   // ...
375   // ...
376   // ...
377   // ...
378   // ...
379   // ...
380   // ...
381   // ...
382   // ...
383   // ...
384   // ...
385   // ...
386   // ...
387   // ...
388   // ...
389   // ...
390   // ...
391   // ...
392   // ...
393   // ...
394   // ...
395   // ...
396   // ...
397   // ...
398   // ...
399   // ...
400   // ...
401   // ...
402   // ...
403   // ...
404   // ...
405   // ...
406   // ...
407   // ...
408   // ...
409   // ...
410   // ...
411   // ...
412   // ...
413   // ...
414   // ...
415   // ...
416   // ...
417   // ...
418   // ...
419   // ...
420   // ...
421   // ...
422   // ...
423   // ...
424   // ...
425   // ...
426   // ...
427   // ...
428   // ...
429   // ...
430   // ...
431   // ...
432   // ...
433   // ...
434   // ...
435   // ...
436   // ...
437   // ...
438   // ...
439   // ...
440   // ...
441   // ...
442   // ...
443   // ...
444   // ...
445   // ...
446   // ...
447   // ...
448   // ...
449   // ...
450   // ...
451   // ...
452   // ...
453   // ...
454   // ...
455   // ...
456   // ...
457   // ...
458   // ...
459   // ...
460   // ...
461   // ...
462   // ...
463   // ...
464   // ...
465   // ...
466   // ...
467   // ...
468   // ...
469   // ...
470   // ...
471   // ...
472   // ...
473   // ...
474   // ...
475   // ...
476   // ...
477   // ...
478   // ...
479   // ...
480   // ...
481   // ...
482   // ...
483   // ...
484   // ...
485   // ...
486   // ...
487   // ...
488   // ...
489   // ...
490   // ...
491   // ...
492   // ...
493   // ...
494   // ...
495   // ...
496   // ...
497   // ...
498   // ...
499   // ...
500   // ...
501   // ...
502   // ...
503   // ...
504   // ...
505   // ...
506   // ...
507   // ...
508   // ...
509   // ...
510   // ...
511   // ...
512   // ...
513   // ...
514   // ...
515   // ...
516   // ...
517   // ...
518   // ...
519   // ...
520   // ...
521   // ...
522   // ...
523   // ...
524   // ...
525   // ...
526   // ...
527   // ...
528   // ...
529   // ...
530   // ...
531   // ...
532   // ...
533   // ...
534   // ...
535   // ...
536   // ...
537   // ...
538   // ...
539   // ...
540   // ...
541   // ...
542   // ...
543   // ...
544   // ...
545   // ...
546   // ...
547   // ...
548   // ...
549   // ...
550   // ...
551   // ...
552   // ...
553   // ...
554   // ...
555   // ...
556   // ...
557   // ...
558   // ...
559   // ...
560   // ...
561   // ...
562   // ...
56
```

OUTPUT

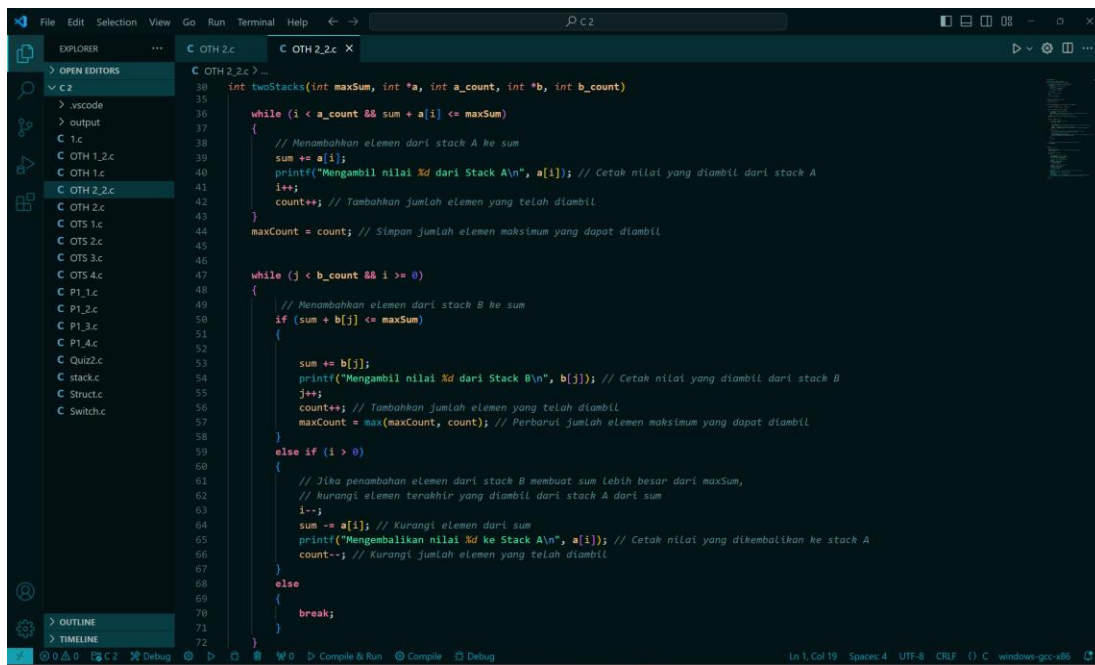
```
PS C:\VsCode\C 2\output> cd 'c:\VsCode\C 2\output'  
PS C:\VsCode\C 2\output> & .\'OTH 2.exe'  
INFORMATIKA  
PS C:\VsCode\C 2\output> █
```

NO 2 STACK

SS dan Penjelasan



```
1 #include <stdio.h>
2
3 // deklarasi fungsi dengan tiga parameter
4 void printStack(int *stack, int count, char *name)
5 {
6     printf("Stack %s : ", name); // cetak nama Stack
7     // cek apakah jumlah element dalam stack lebih dari 0
8     if (count > 0)
9     {
10        // jika iya maka cetak seperti dibawah
11        printf("[ ");
12        for (int i = 0; i < count; i++)
13        {
14            printf("%d ", stack[i]);
15        }
16        printf("]");
17    }
18    // jika tidak maka cetak EMPTY
19    else
20    {
21        printf("[ ] ( Empty )");
22    }
23 }
24
25 // fungsi untuk membandingkan dua bilangan
26 int max(int a, int b)
27 {
28     return (a > b) ? a : b;
29 }
30
31 int twoStacks(int maxSum, int *a, int a_count, int *b, int b_count)
32 {
33     // Inisialisasi variabel
34     int i = 0, j = 0, sum = 0, count = 0, maxCount = 0;
35
36     while (i < a_count && sum + a[i] <= maxSum)
37     {
38        // Menambahkan elemen dari stack A ke sum
39        sum += a[i];
40        printf("Mengambil nilai %d dari Stack A\n", a[i]); // Cetak nilai yang diambil dari stack A
41        i++;
42        count++; // Tambahkan jumlah elemen yang telah diambil
43    }
44    maxCount = count; // Simpan jumlah elemen maksimum yang dapat diambil
45
46    while (j < b_count && i >= 0)
47    {
48        // Menambahkan elemen dari stack B ke sum
49        if (sum + b[j] <= maxSum)
50        {
51            sum += b[j];
52            printf("Mengambil nilai %d dari Stack B\n", b[j]); // Cetak nilai yang diambil dari stack B
53            j++;
54            count++; // Tambahkan jumlah elemen yang telah diambil
55            maxCount = max(maxCount, count); // Perbarui jumlah elemen maksimum yang dapat diambil
56        }
57        else if (i > 0)
58        {
59            // Jika penambahan elemen dari stack B membuat sum lebih besar dari maxSum,
60            // kurangi elemen terakhir yang diambil dari stack A dari sum
61            i--;
62            sum -= a[i]; // Kurangi elemen dari sum
63            printf("Mengembalikan nilai %d ke Stack A\n", a[i]); // Cetak nilai yang dikembalikan ke stack A
64            count--; // Kurangi jumlah elemen yang telah diambil
65        }
66        else
67        {
68            break;
69        }
70    }
71 }
72
```



```
30 int twoStacks(int maxSum, int *a, int a_count, int *b, int b_count)
31 {
32     while (i < a_count && sum + a[i] <= maxSum)
33     {
34        // Menambahkan elemen dari stack A ke sum
35        sum += a[i];
36        printf("Mengambil nilai %d dari Stack A\n", a[i]); // Cetak nilai yang diambil dari stack A
37        i++;
38        count++; // Tambahkan jumlah elemen yang telah diambil
39    }
40    maxCount = count; // Simpan jumlah elemen maksimum yang dapat diambil
41
42    while (j < b_count && i >= 0)
43    {
44        // Menambahkan elemen dari stack B ke sum
45        if (sum + b[j] <= maxSum)
46        {
47            sum += b[j];
48            printf("Mengambil nilai %d dari Stack B\n", b[j]); // Cetak nilai yang diambil dari stack B
49            j++;
50            count++; // Tambahkan jumlah elemen yang telah diambil
51            maxCount = max(maxCount, count); // Perbarui jumlah elemen maksimum yang dapat diambil
52        }
53        else if (i > 0)
54        {
55            // Jika penambahan elemen dari stack B membuat sum lebih besar dari maxSum,
56            // kurangi elemen terakhir yang diambil dari stack A dari sum
57            i--;
58            sum -= a[i]; // Kurangi elemen dari sum
59            printf("Mengembalikan nilai %d ke Stack A\n", a[i]); // Cetak nilai yang dikembalikan ke stack A
60            count--; // Kurangi jumlah elemen yang telah diambil
61        }
62        else
63        {
64            break;
65        }
66    }
67 }
68
```


OUTPUT

```
PS C:\VsCode\C 2\output> cd 'c:\VsCode\C 2\output'
PS C:\VsCode\C 2\output> & .\OTH 2_2.exe'
Input Jumlah Games: 1
Input Stack A, Stack B, maxSum: 5 4 11
Input Element Stack A: 4 5 2 1 1
Input Element Stack B: 3 1 1 2

-----
-----
Stack A : [ 4 5 2 1 1 ]
Stack B : [ 3 1 1 2 ]

Mengambil nilai 4 dari Stack A
Mengambil nilai 5 dari Stack A
Mengambil nilai 2 dari Stack A
Mengembalikan nilai 2 ke Stack A
Mengembalikan nilai 5 ke Stack A
Mengambil nilai 3 dari Stack B
Mengambil nilai 1 dari Stack B
Mengambil nilai 1 dari Stack B
Mengambil nilai 2 dari Stack B

Output: 5
-----
-----
PS C:\VsCode\C 2\output> 
```