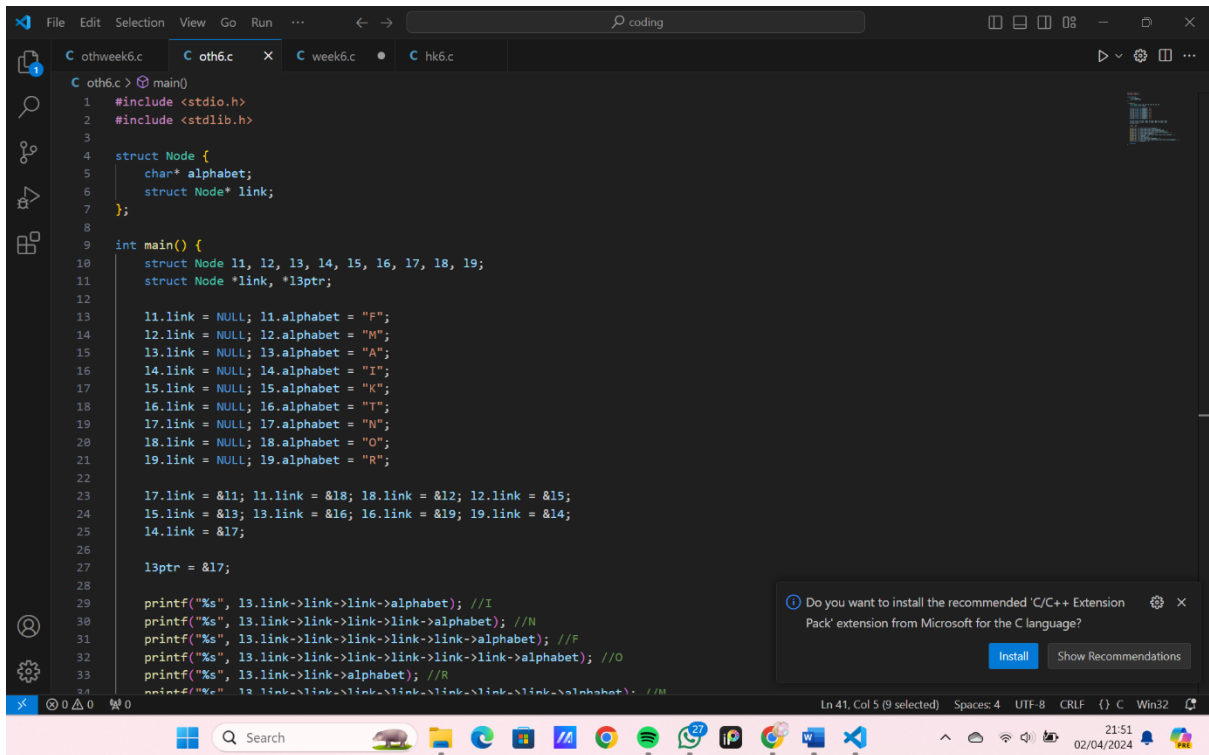


Nama : MARIA ROSA WAHYUNING UTAMI

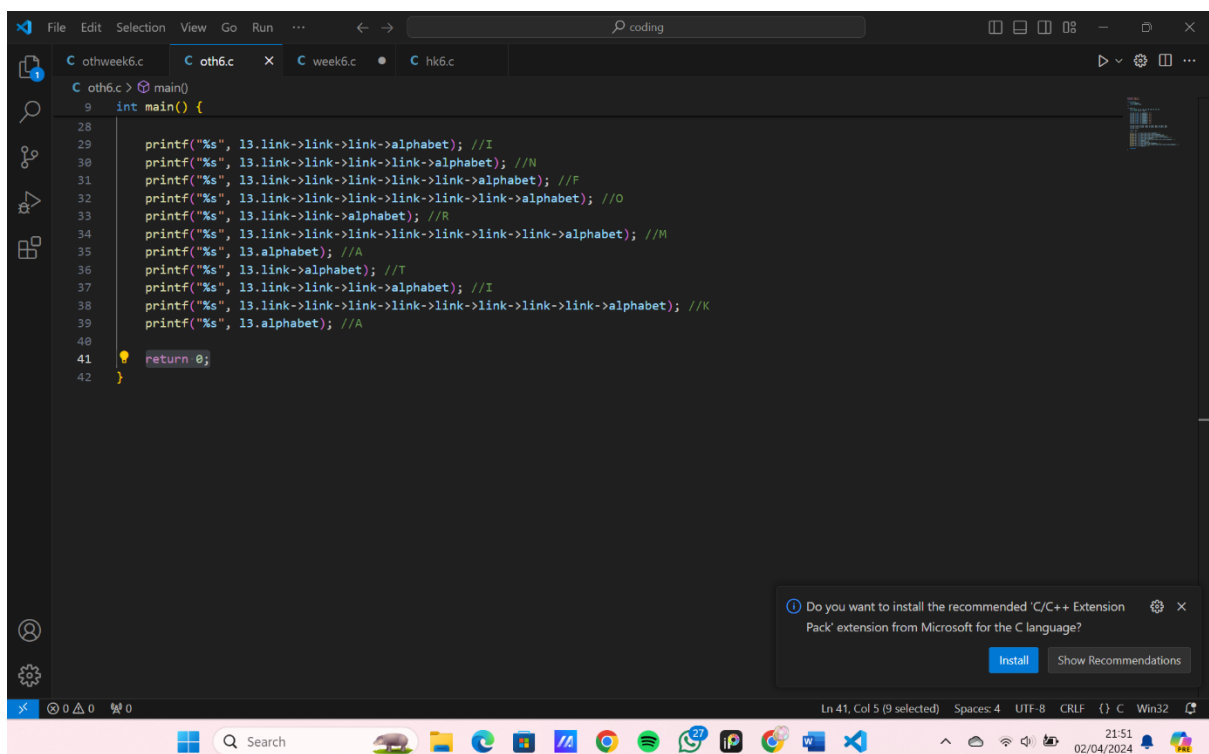
NIM : 1203230123

## TUGAS OTH WEEK 6

1. Asisten Sherlock Holmes
  - a. Source Code

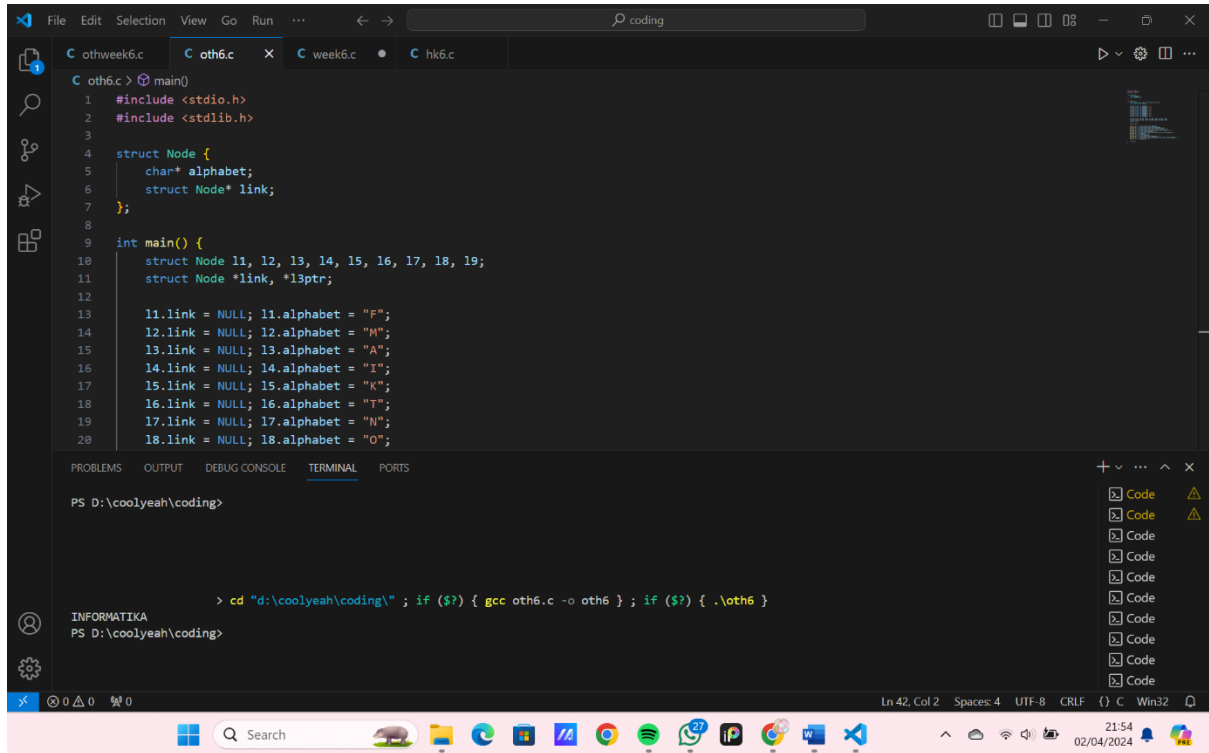


```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     char* alphabet;
6     struct Node* link;
7 };
8
9 int main() {
10     struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
11     struct Node *link, *l3ptr;
12
13     l1.link = NULL; l1.alphabet = "F";
14     l2.link = NULL; l2.alphabet = "M";
15     l3.link = NULL; l3.alphabet = "A";
16     l4.link = NULL; l4.alphabet = "I";
17     l5.link = NULL; l5.alphabet = "K";
18     l6.link = NULL; l6.alphabet = "T";
19     l7.link = NULL; l7.alphabet = "N";
20     l8.link = NULL; l8.alphabet = "O";
21     l9.link = NULL; l9.alphabet = "R";
22
23     l7.link = &l1; l1.link = &l8; l8.link = &l2; l2.link = &l5;
24     l5.link = &l3; l3.link = &l6; l6.link = &l9; l9.link = &l4;
25     l4.link = &l7;
26
27     l3ptr = &l7;
28
29     printf("%s", l3.link->link->link->alphabet); //I
30     printf("%s", l3.link->link->link->link->alphabet); //N
31     printf("%s", l3.link->link->link->link->link->alphabet); //F
32     printf("%s", l3.link->link->link->link->link->link->alphabet); //O
33     printf("%s", l3.link->link->alphabet); //R
34     printf("%s", l3.link->link->link->link->link->link->link->link->alphabet); //M
```



```
28     printf("%s", l3.link->link->link->alphabet); //I
29     printf("%s", l3.link->link->link->link->alphabet); //N
30     printf("%s", l3.link->link->link->link->link->alphabet); //F
31     printf("%s", l3.link->link->link->link->link->link->alphabet); //O
32     printf("%s", l3.link->link->alphabet); //R
33     printf("%s", l3.link->link->link->link->link->link->alphabet); //M
34     printf("%s", l3.alphabet); //A
35     printf("%s", l3.link->alphabet); //T
36     printf("%s", l3.link->link->link->alphabet); //I
37     printf("%s", l3.link->link->link->link->link->link->link->link->alphabet); //K
38     printf("%s", l3.alphabet); //A
39
40     return 0;
41 }
```

## b. Output



```
File Edit Selection View Go Run ...
C othweek6.c C oth6.c x week6.c hk6.c
C oth6.c main()
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     char* alphabet;
6     struct Node* link;
7 };
8
9 int main() {
10     struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
11     struct Node *link, *l3ptr;
12
13     l1.link = NULL; l1.alphabet = "F";
14     l2.link = NULL; l2.alphabet = "M";
15     l3.link = NULL; l3.alphabet = "A";
16     l4.link = NULL; l4.alphabet = "I";
17     l5.link = NULL; l5.alphabet = "K";
18     l6.link = NULL; l6.alphabet = "T";
19     l7.link = NULL; l7.alphabet = "N";
20     l8.link = NULL; l8.alphabet = "O";
21
22     printf("Linked List: ");
23     while (l1 != NULL) {
24         printf("%s ", l1->alphabet);
25         l1 = l1->link;
26     }
27     printf("\n");
28 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\coolyeah\coding>
> cd "d:\coolyeah\coding\" ; if ($?) { gcc oth6.c -o oth6 } ; if ($?) { .\oth6 }
INFORMATIKA
PS D:\coolyeah\coding>
```

## c. Penjelasan

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

Baris pertama adalah **stdio.h** yang merupakan deklarasi library untuk proses input-output standar, dan **stdlib.h** untuk alokasi memori dinamis

```
struct Node {
    char* alphabet;
    struct Node* link;
};
```

Mendeklarasikan struktur **Node** untuk membuat jenis data baru. Dengan **alphabet** yang merupakan pointer ke karakter (huruf) dan **link** yang merupakan pointer ke struktur **Node** berikutnya dalam linked list. **link** akan menunjuk ke simpul berikutnya dalam rangkaian

```
int main() {
    struct Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
    struct Node *link, *l3ptr;

    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 = "P";
}
```

```

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";

```

Memasuki fungsi utama program, ada beberapa variabel dengan tipe **struct Node** dideklarasikan untuk merepresentasikan beberapa simpul dalam linked list yang berjumlah 9 simpul. Setiap simpul diinisialisasi dengan **link** yang belum terhubung ke apapun (**NULL**) dan **alphabet** yang berisi huruf tertentu

```

17.link = &11; 11.link = &18; 18.link = &12; 12.link = &15;
15.link = &13; 13.link = &16; 16.link = &19; 19.link = &14;
14.link = &17;

```

Setelah inisialisasi, koneksi antara simpul-simpul tadi diatur membentuk linked list yang diinginkan dengan mengatur pointer link pada setiap simpul untuk menunjuk ke simpul berikutnya dalam urutan yang diinginkan. Misalnya 17 dihubungkan ke 11, 11 dihubungkan ke 18, dan seterusnya hingga Kembali ke 17

```
13ptr = &17;
```

Deklarasi variabel **13ptr** untuk menunjukkan titik awal dalam linked list yang menunjuk ke simpul berisi huruf N

```

printf("%s", 13.link->link->link->alphabet); //I
printf("%s", 13.link->link->link->link->alphabet); //N
printf("%s", 13.link->link->link->link->link->alphabet); //F
printf("%s", 13.link->link->link->link->link->link->alphabet); //O
printf("%s", 13.link->link->alphabet); //R
printf("%s", 13.link->link->link->link->link->link->link->alphabet); //M
printf("%s", 13.alphabet); //A
printf("%s", 13.link->alphabet); //T
printf("%s", 13.link->link->link->alphabet); //I
printf("%s", 13.link->link->link->link->link->link->link->link->alphabet); //K
printf("%s", 13.alphabet); //A

```

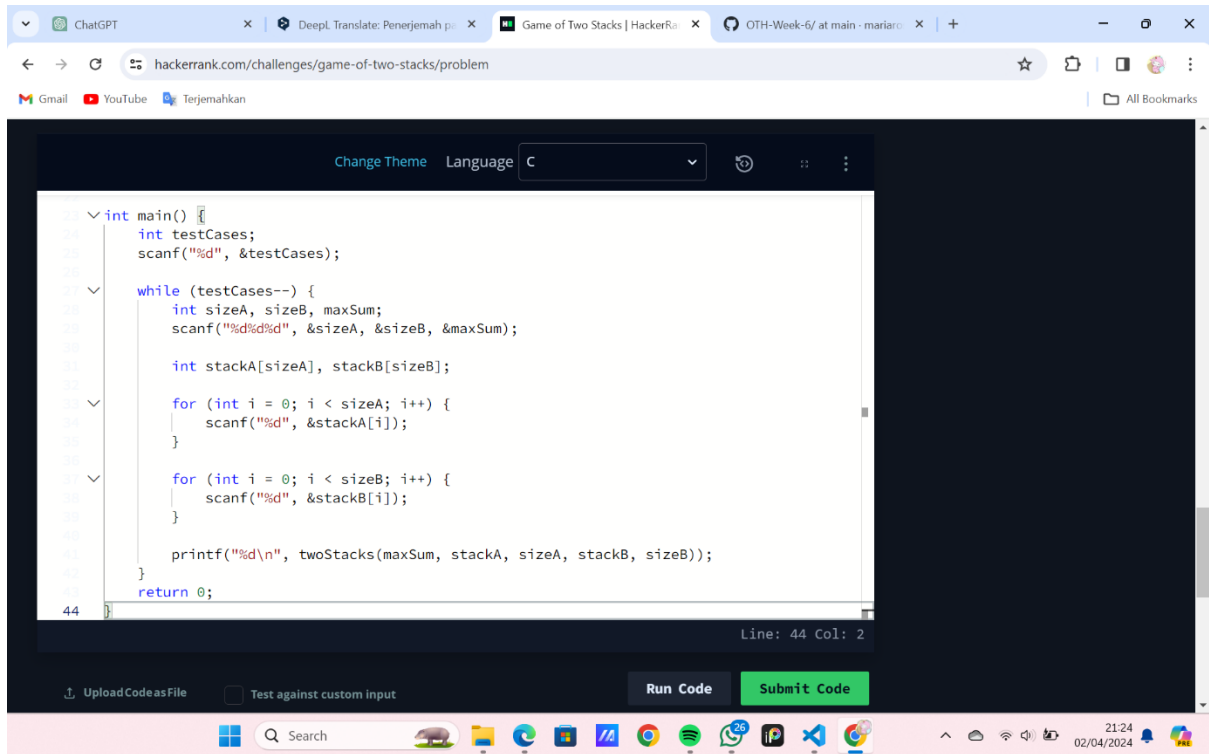
Program tersebut bekerja dengan cara mengakses simpul-simpul dalam struktur linked list yang telah dibuat dan kemudian mencetak huruf yang terkandung dalam setiap simpulnya, dimulai dari simpul 13.

```
return 0;
```

Menunjukkan bahwa program telah berhasil dijalankan dan berakhir dengan sukses tanpa ada kesalahan

## 2. Hackerrank

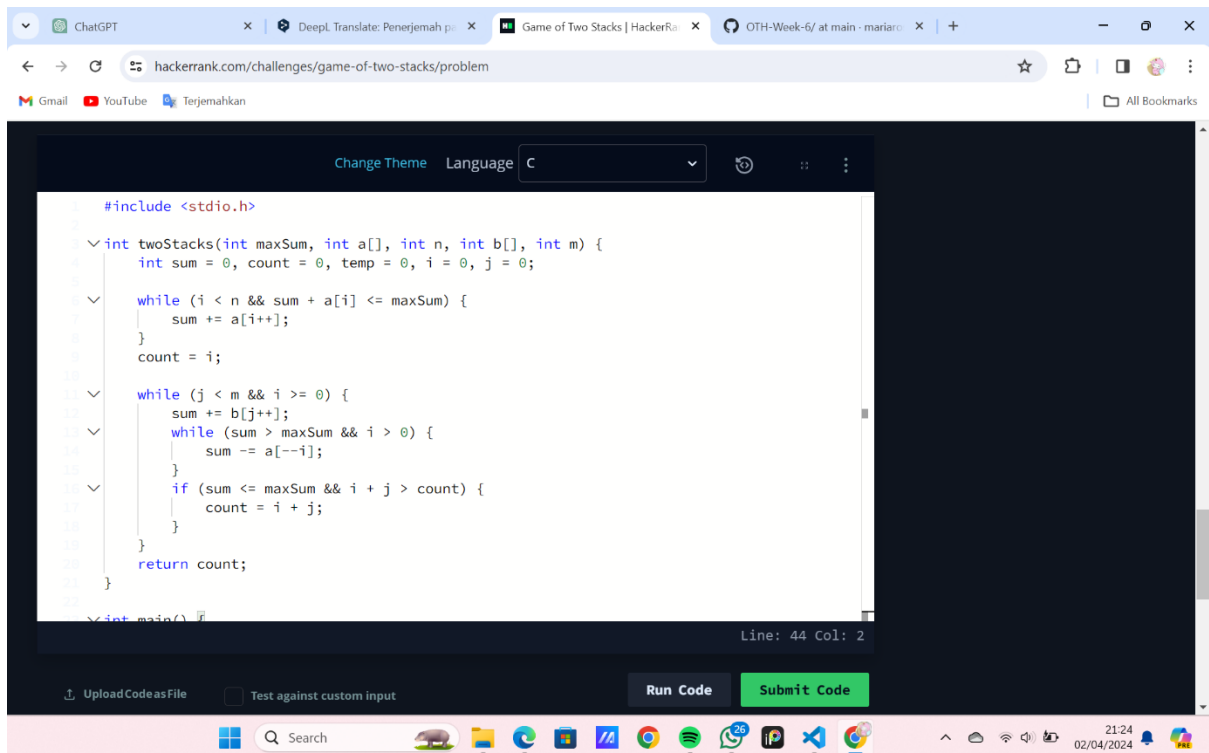
### a. Source Code



```
1  int main() {  
2      int testCases;  
3      scanf("%d", &testCases);  
4  
5      while (testCases--) {  
6          int sizeA, sizeB, maxSum;  
7          scanf("%d%d%d", &sizeA, &sizeB, &maxSum);  
8  
9          int stackA[sizeA], stackB[sizeB];  
10  
11         for (int i = 0; i < sizeA; i++) {  
12             scanf("%d", &stackA[i]);  
13         }  
14  
15         for (int i = 0; i < sizeB; i++) {  
16             scanf("%d", &stackB[i]);  
17         }  
18  
19         printf("%d\n", twoStacks(maxSum, stackA, sizeA, stackB, sizeB));  
20     }  
21     return 0;  
22 }
```

Line: 44 Col: 2

Upload Code as File Test against custom input Run Code Submit Code



```
1  #include <stdio.h>  
2  
3  int twoStacks(int maxSum, int a[], int n, int b[], int m) {  
4      int sum = 0, count = 0, temp = 0, i = 0, j = 0;  
5  
6      while (i < n && sum + a[i] <= maxSum) {  
7          sum += a[i++];  
8      }  
9      count = i;  
10  
11      while (j < m && i >= 0) {  
12          sum += b[j++];  
13          while (sum > maxSum && i > 0) {  
14              sum -= a[--i];  
15          }  
16          if (sum <= maxSum && i + j > count) {  
17              count = i + j;  
18          }  
19      }  
20      return count;  
21  }  
22  
23  int main() {  
24      int testCases;  
25      scanf("%d", &testCases);  
26  
27      while (testCases--) {  
28          int sizeA, sizeB, maxSum;  
29          scanf("%d%d%d", &sizeA, &sizeB, &maxSum);  
30  
31          int stackA[sizeA], stackB[sizeB];  
32  
33          for (int i = 0; i < sizeA; i++) {  
34              scanf("%d", &stackA[i]);  
35          }  
36  
37          for (int i = 0; i < sizeB; i++) {  
38              scanf("%d", &stackB[i]);  
39          }  
40  
41          printf("%d\n", twoStacks(maxSum, stackA, sizeA, stackB, sizeB));  
42      }  
43      return 0;  
44  }
```

Line: 44 Col: 2

Upload Code as File Test against custom input Run Code Submit Code

## b. Output/Hasil

The screenshot shows the HackerRank interface for the 'Game of Two Stacks' problem. A green banner at the top says 'Congratulations' and 'You solved this challenge. Would you like to challenge your friends?'. A 'Next Challenge' button is visible. On the left, a list of test cases from 0 to 6 is shown, all with green checkmarks indicating they passed. The main area displays the 'Compiler Message' as 'Success'. Below this, the 'Input (stdin)' is shown as a 4x4 grid of numbers: 

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

. The 'Expected Output' is a single line with the number 4. The Windows taskbar at the bottom shows the time as 21:21 on 02/04/2024.

This screenshot shows the same HackerRank page but with the code submission area visible. At the top, there are buttons for 'Run Code' and 'Submit Code'. Below these, the same test case list and input/output details are shown. The Windows taskbar at the bottom shows the time as 21:24 on 02/04/2024.

### c. Penjelasan

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

Deklarasi library standar untuk input-output

```
int twoStacks(int maxSum, int a[], int n, int b[], int m) {  
    int sum = 0, count = 0, temp = 0, i = 0, j = 0;
```

Deklarasi fungsi **twoStacks** yang mengambil parameter **maxSum** untuk batas maksimum jumlah dari kedua stack, **a[]** dan **b[]** yakni array yang merepresentasikan stack A dan stack B, serta **n** dan **m** yang merupakan jumlah elemen dalam stack A dan stack B

```
while (i < n && sum + a[i] <= maxSum) {  
    sum += a[i++];  
}  
count = i;
```

Loop while yang akan berjalan selama **i** masih kurang dari jumlah elemen dalam stack A (**n**) dan penambahan nilai **sum** dengan elemen ke-**i** dari stack A (**a[i]**) masih kurang dari atau sama dengan **maxSum**. Di dalam loop, nilai **sum** akan bertambah dengan nilai elemen ke-**i** dari stack A (**a[i]**) dan **i** akan bertambah satu setiap kali loop dieksekusi dengan menggunakan operator **++**. Loop ini bertujuan untuk menemukan banyaknya elemen yang dapat diambil dari stack A sehingga jumlahnya tidak melebihi **maxSum**. Setelah loop pertama selesai, nilai dari **count** akan menjadi jumlah elemen yang dapat diambil dari stack A tanpa melebihi **maxSum**

```
while (j < m && i >= 0) {  
    sum += b[j++];
```

Ini adalah loop while luar yang akan terus berjalan selama belum semua elemen dalam stack B telah dicoba (**j < m**) dan masih ada elemen yang dapat diambil dari stack A (**i >= 0**). Loop ini memiliki fungsi untuk mencoba semua kemungkinan pengambilan elemen dari kedua stack. Nilai elemen ke-**j** dari stack B akan ditambahkan ke **sum** dan pada saat yang sama menaikkan nilai **j** dengan 1 menggunakan operator **++**

```
while (sum > maxSum && i > 0) {  
    sum -= a[--i];
```

Ini adalah loop while dalam loop while luar yang akan berjalan selama nilai **sum** melebihi **maxSum** dan masih ada elemen yang telah diambil dari stack A (**i > 0**). Loop ini bertujuan mengurangi elemen-elemen yang telah diambil dari stack A sehingga nilai **sum** tidak melebihi **maxSum**. Kemudian nilai elemen ke-**(i-1)** dari stack A dari **sum** akan dikurangi dan menurunkan nilai **i** dengan 1 menggunakan operator **-**

```
if (sum <= maxSum && i + j > count) {  
    count = i + j;
```

Pernyataan kondisional yang akan dievaluasi jika setelah melakukan langkah-langkah di dalam loop, nilai **sum** tidak melebihi **maxSum** dan jumlah elemen-elemen yang telah diambil dari stack A dan stack B (**i + j**) lebih besar dari nilai **count**. Jika kondisi terpenuhi maka nilai **count** akan diperbarui dengan nilai **i + j**

```
return count;
```

Setelah loop selesai, fungsi **twoStacks** akan mengembalikan nilai **count** sebagai hasil akhir dari berapa banyak elemen yang dapat diambil dari kedua stack tanpa melebihi **maxSum**

```
int main() {
```

Titik masuk utama program

```
int testCases;
```

```
scanf("%d", &testCases);
```

Program ini akan membaca jumlah kasus uji yang akan dieksekusi dengan menggunakan variabel **testCases**. Fungsi **scanf** untuk membaca jumlah uji kasus dari input pengguna dan menyimpannya ke variabel **testCases**

```
while (testCases--)
```

Loop while yang akan dieksekusi sebanyak **testCases** kali. Loop ini akan terus berjalan hingga semua kasus uji telah dievaluasi

```
int sizeA, sizeB, maxSum;
```

```
scanf("%d%d%d", &sizeA, &sizeB, &maxSum);
```

Deklarasi variabel **sizeA** untuk menyimpan ukuran stack A, **sizeB** untuk ukuran stack B, dan **maxSum** untuk nilai maksimum yang diizinkan. Fungsi **scanf** digunakan untuk membaca dan menyimpannya ke variabel terkait secara berturut-turut

```
int stackA[sizeA], stackB[sizeB];
```

deklarasi array **stackA** dan **stackB** dengan ukuran yang sesuai berdasarkan nilai yang dibaca sebelumnya untuk menyimpan elemen-elemen stack A dan B

```
for (int i = 0; i < sizeA; i++) {
```

```
scanf("%d", &stackA[i]);
```

Loop for untuk membaca nilai-nilai elemen dari stack A yang diinput dan menyimpannya dalam array **stackA**

```
for (int i = 0; i < sizeB; i++) {
```

```
scanf("%d", &stackB[i]);
```

Loop for untuk elemen stack B dengan fungsi yang sama seperti loop elemen stack A

```
printf("%d\n", twoStacks(maxSum, stackA, sizeA, stackB, sizeB));
```

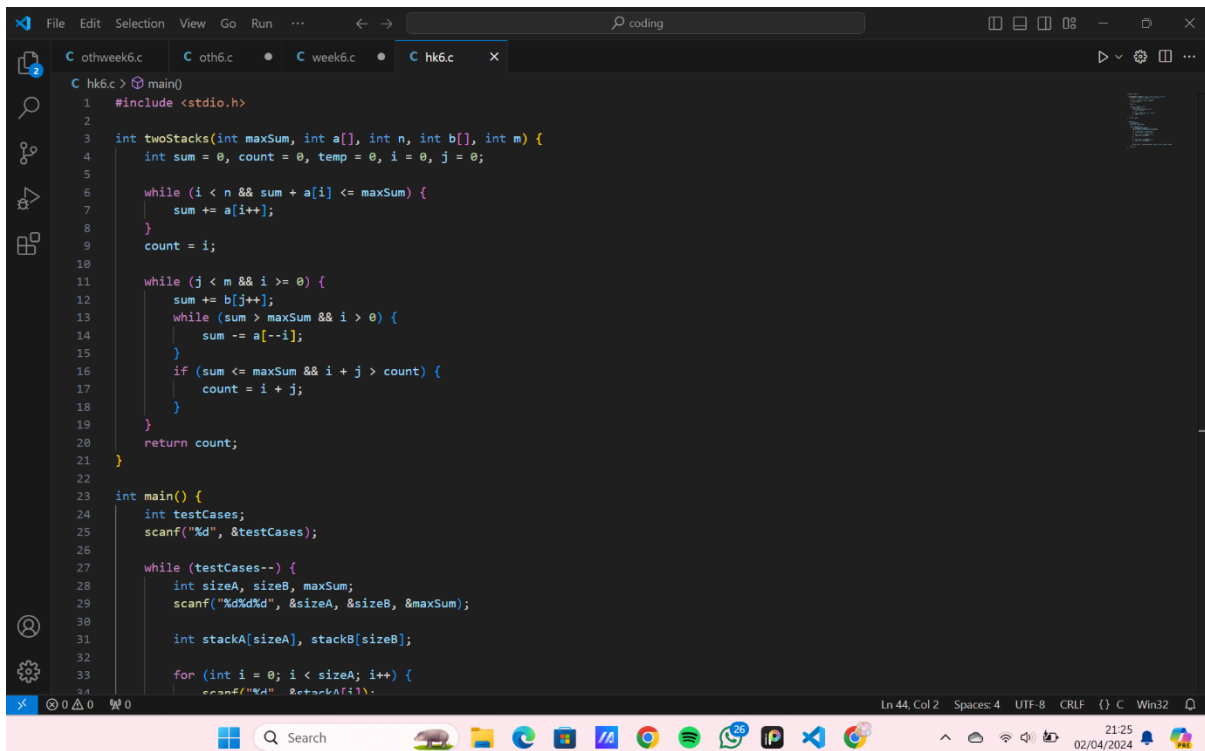
memanggil fungsi **twoStacks** untuk menghitung jumlah maksimum elemen yang dapat diambil dari kedua stack tanpa melebihi nilai maksimum. Hasil perhitungan tersebut akan dicetak ke layar

```
return 0;
```

Menandakan program berakhir dengan sukses

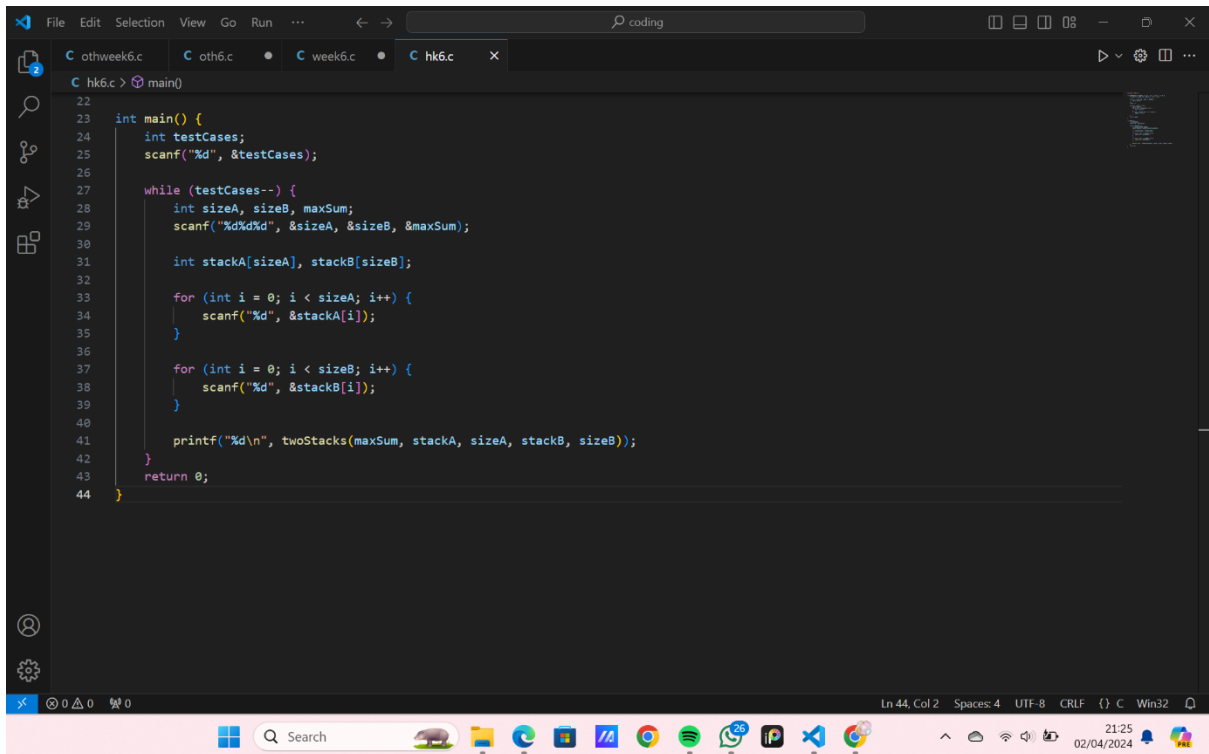
### 3. Soal hackerrank versi vs code

#### a. Source Code



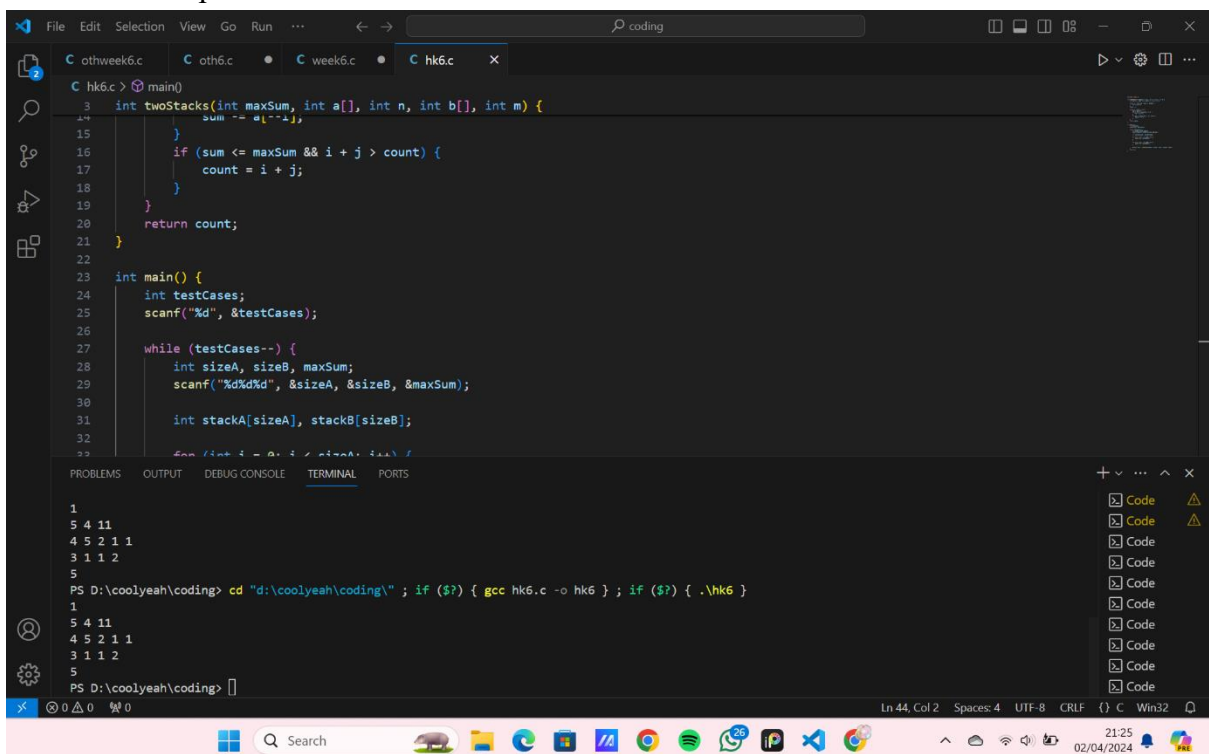
```
C h6.c > main()
1  #include <stdio.h>
2
3  int twoStacks(int maxSum, int a[], int n, int b[], int m) {
4      int sum = 0, count = 0, temp = 0, i = 0, j = 0;
5
6      while (i < n && sum + a[i] <= maxSum) {
7          sum += a[i++];
8      }
9      count = i;
10
11     while (j < m && i >= 0) {
12         sum += b[j++];
13         while (sum > maxSum && i > 0) {
14             sum -= a[--i];
15         }
16         if (sum <= maxSum && i + j > count) {
17             count = i + j;
18         }
19     }
20     return count;
21 }
22
23 int main() {
24     int testCases;
25     scanf("%d", &testCases);
26
27     while (testCases--) {
28         int sizeA, sizeB, maxSum;
29         scanf("%d%d%d", &sizeA, &sizeB, &maxSum);
30
31         int stackA[sizeA], stackB[sizeB];
32
33         for (int i = 0; i < sizeA; i++) {
34             scanf("%d", &stackA[i]);
35         }
36         for (int j = 0; j < sizeB; j++) {
37             scanf("%d", &stackB[j]);
38         }
39
40         printf("%d\n", twoStacks(maxSum, stackA, sizeA, stackB, sizeB));
41     }
42 }
```





```
22
23 int main() {
24     int testCases;
25     scanf("%d", &testCases);
26
27     while (testCases--) {
28         int sizeA, sizeB, maxSum;
29         scanf("%d%d", &sizeA, &sizeB, &maxSum);
30
31         int stackA[sizeA], stackB[sizeB];
32
33         for (int i = 0; i < sizeA; i++) {
34             scanf("%d", &stackA[i]);
35         }
36
37         for (int i = 0; i < sizeB; i++) {
38             scanf("%d", &stackB[i]);
39         }
40
41         printf("%d\n", twoStacks(maxSum, stackA, sizeA, stackB, sizeB));
42     }
43     return 0;
44 }
```

## b. Output



```
3 int twoStacks(int maxSum, int a[], int n, int b[], int m) {
4     int sum = 0, count = 0;
5
6     for (int i = 0; i < n; i++) {
7         sum += a[i];
8         if (sum > maxSum) {
9             sum = 0;
10            count++;
11        }
12    }
13
14    for (int i = 0; i < m; i++) {
15        sum += b[i];
16        if (sum > maxSum) {
17            sum = 0;
18            count++;
19        }
20    }
21    return count;
22 }
23
24 int main() {
25     int testCases;
26     scanf("%d", &testCases);
27
28     while (testCases--) {
29         int sizeA, sizeB, maxSum;
30         scanf("%d%d", &sizeA, &sizeB, &maxSum);
31
32         int stackA[sizeA], stackB[sizeB];
33
34         for (int i = 0; i < sizeA; i++) {
35             scanf("%d", &stackA[i]);
36         }
37
38         for (int i = 0; i < sizeB; i++) {
39             scanf("%d", &stackB[i]);
40         }
41
42         printf("%d\n", twoStacks(maxSum, stackA, sizeA, stackB, sizeB));
43     }
44     return 0;
45 }
```

```
1
2 5 4 11
3 4 5 2 1 1
4 3 1 1 2
5
PS D:\coolyeah\coding> cd "d:\coolyeah\coding\" ; if ($?) { gcc hk6.c -o hk6 }; if ($?) { .\hk6 }
1
2 5 4 11
3 4 5 2 1 1
4 3 1 1 2
5
PS D:\coolyeah\coding>
```