

## Mastery I — Data Struct. & Algo. (T. III/18–19)

Name: \_\_\_\_\_

ID: \_\_\_\_\_

### Directions:

- You have 110 minutes (i.e., 1 hour and 50 minutes) to complete the following examination.
- There are 4 problems. The maximum possible score is 35. We will grade you out of  $T \leq 30$ , where  $T$  is yet to be decided. Anything above  $T$  is extra credit. This means, you should think of it as *three* real problems plus *one* extra credit.
- No collaboration of any kind whatsoever is permitted during the exam.
- **WHAT IS PERMITTED:**
  - Reading the official Java documentation
  - Accessing Canvas for submission.
- **WHAT IS NOT PERMITTED:**
  - Browsing (online) tutorials or reading stack overflow threads.
  - Accessing previously-written code on your own machine.
  - Communicating with other person or using any other aid.
- For each problem, the entirety of your solution must live in one file, named according to the instructions in this handout. When grading a problem, the script will only compile that one file for the problem. **Importantly:** your implementation must not be part of a package.
- We're providing a starter package, which you can download at  
`https://cs.muic.mahidol.ac.th/courses/ds/malkist.zip`  
The password is “crunchy”.  
When you unpack the package, you'll see one file for each problem.
- To submit your work, zip all your Java files as one zip file called `mastery1.zip` and upload it to Canvas.

1	2	3	4	$\Sigma$

## Problem 1: Best Split (10 points)

The input to this problem is an array `a[]` of arbitrarily-ordered integers. Your task is to find the index  $m$ ,  $0 \leq m \leq a.length$  that minimizes  $(\text{sum}(a[:m]) - \text{sum}(a[m:]))^2$ . That is, it minimizes the square of the difference between the sums of the two sides.

Inside `BestSplit.java`, you will implement a function

```
public static int bestSplit(int[] a)
```

that takes in an array `a[]` of integers and returns an `int` representing the index that minimizes the difference described above.

*Sample Input/Output:*

- `bestSplit([8, 4, 5])` should return 1. This is, the best split yields `[8]` vs. `[4, 5]`, where the square of the difference is  $(8 - 9)^2 = 1$ .
- `bestSplit([3, 1, 4, 5, 9, 2, 6, 5, 3, 5, 8, 7, 3, 11])` should return 8, where the square of the difference is  $(35 - 37)^2 = 4$ .

### Constraints & Grading:

- The input array contains at least 2 elements and at most 20,000,000 elements.
- We aren't trying to be malicious, but you should know that if you square an `int`, the result doesn't always fit in an `int`—consider an alternative, e.g., using `long`.
- Your solution must finish within 2 seconds per test. The desired solution must run in  $O(n)$  time. Slower solutions will receive some partial credit.

## Problem 2: Lost Items (10 points)

Dr. Piti keeps a rare collection of numbers. In this collection, some numbers are repeated multiple times. He treasures this collection so much that he keeps two identical copies of the collection,  $a$  and  $b$  (so they are backups of each other).

One day, he took the collection  $a$  to a Deadly Math class, and some numbers were lost; some kids took them after watching card tricks. The collection is rather large, so he asked you to help him identify the missing numbers.

Inside `LostItems.java`, you will write a function

```
public static int[] lostItems(int[] a, int b[])
```

that takes in both  $a$  and  $b$ , each an array of `ints`, and returns an array of `ints` containing all the lost numbers. He has made the following requests:

- From his perspective, a number  $x$  is lost if the number of times  $x$  appears in  $a$  is less than the number of times it appears in  $b$ . For example, if 203 appears 3 times in  $b$  but only once in  $a$ , the number 203 is lost—two copies have disappeared.
- Each lost number is reported in the output only once, even if multiple copies are lost.
- The output array must be ordered from small to large.

## Promises, Constraints, and Grading

- $1 \leq a.length \leq b.length \leq 100,000$ .
- We promise that every number in  $a$  appears in  $b$ .
- Each  $b[i]$  satisfies  $0 \leq b[i] \leq 16,384$ . (*Hint*: it is possible to keep an `int` array of length 16,384 or more. For example, `int[] counts = new int[16385];`)
- Your solution must finish within 2 seconds per test. Let  $n$  be the length of  $b$ . The desired solution must run in  $O(n \log n)$  time or faster. (*Hint*: Both  $O(n \log n)$  and  $O(n)$  solutions are within reach.)
- Partial credit will be given to  $O(n^2)$  solutions.




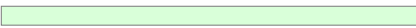
*Example:*

```
a = {203, 204, 205, 206, 207, 208, 203, 204, 205, 206}
b = {203, 204, 204, 205, 206, 207, 205, 208, 203, 206, 205, 206, 204}
lostItems(a, b) should return {204, 205, 206}.
```

## Problem 3: Colorful Sticks (10 points)

Gift and K2 own a company that produces sticks. The company only produces sticks in three lengths: 3 meters, 4 meters, and 11 meters. For distinctive look and feel, 3-meter sticks come in red, 4-meter sticks come in blue, and 11-meter sticks come in green.

In this problem, we're interested in the number of possible patterns that an  $n$ -meter segment can be made out of these sticks. For instance, we can make a segment of length 11 meters in 4 patterns, as follows:

- Using 3-meter, 4-meter, 4-meter sticks: 
- Using 4-meter, 3-meter, 4-meter sticks: 
- Using 4-meter, 4-meter, 3-meter sticks: 
- Using an 11-meter stick: 

Inside `Weave.java`, you'll write a function

```
public static int numWeaves(int n)
```

that takes  $n$  as input and returns an integer representing the number of patterns ("weaves") we can make for a segment of length  $n$  out of these sticks.

## Promises, Constraints, and Grading

- $1 \leq n \leq 75$ . On any  $n$  within this range, your program must run within 2 seconds.
- The number of patterns will fit in an `int`.
- Partial credits will be given to slower code that only solves the problem for small  $n$ .

*Example:*

- `numWeaves(3)==1`
- `numWeaves(6)==1`
- `numWeaves(14)==8`
- `numWeaves(75)==5217077`

## Problem 4: Closest Squares (5 points)

You are given an array  $a[]$  of arbitrarily-ordered integers. It is guaranteed that the integers are distinct. Your task is to find the smallest absolute difference between the squares of two different array elements. Mathematically, you're to find the smallest  $|(a[i])^2 - (a[j])^2|$  with  $i \neq j$ .

Inside `ClosestSquares.java`, you will implement a function

```
public static long closestSquares(int[] a)
```

that takes in an array  $a[]$  of integers and returns an **long** representing the smallest absolute difference between the squares of any two different array elements.

*Sample Input:*

- `closestSquares([3, 5, 2, 7, 1])` should return 3. This is attained by  $2^2 - 1^2 = 4 - 1 = 3$ .
- `closestSquares([5, 9, 1, 11, 2, -9])` should return 0, obtained from  $(9)^2 - (-9)^2 = 0$ .
- `closestSquares([-20, -3916237, -357920, -3620601, 7374819, -7330761, 30, 6246457, -6461594, 266854])` should return 500, obtained from  $(30)^2 - (-20)^2 = 500$ .

### Constraints & Grading:

- The input array contains at least 2 elements and at most 200,000 elements.
- Each  $a[i]$  satisfies  $-10^7 \leq a[i] \leq 10^7$ . This means  $(a[i])^2$  **might be too large to keep in an int**. Consider using the **long** data type.
- Your solution must finish within 1 second per test. The desired solution must run in  $O(n \log n)$  time or faster.
- If your solution runs in  $n^2$  time, you will receive some partial credit.