Programming Assignment 2 (Due Date: 09 April 2021)

For this programming assignment, you will not have to submit any written answers. Instead, you will have to submit a program written in Java or C++11 on CodeCrunch at `https://codecrunch.comp.nus.edu.sg/`. The portal will stop accepting submissions after 09 April 2021 2359h so please start your assignment **early**.

Templates will be provided for all the problems. These templates provide a starting point for your implementation. It is highly recommended to use all the templates given to you. **Do not change the file name or the class name of the template, or else your code may be marked as incorrect**. However, you have to submit your own work. **Posting the question or solution in public repositories are not allowed also counts as a form of plagiarism.** Any form of plagiarism is subject to disciplinary action.

Please include a brief description of your algorithm as a comment at the top of the code as this will be used for marking. You should also explain the time complexity of your algorithm in this comment. **Failure to do so may result in deducted marks.**. It also helps you to gain partial credits if your code has some bugs.

There are some example test cases provided in the assignment folder and these will also be uploaded onto CodeCrunch. If your program fails any of the example test cases, you will get **ZERO** marks. If your program passes all the example test cases, your code will be marked manually so please code neatly and add comments where appropriate. You may be asked to explain your code if the marker cannot understand it. Marks will be deducted if there are bugs in your code or if your algorithm does not meet the time complexity stated in the question. You are strongly encouraged to design your own test cases to test your code.

Additional test cases will be added during marking.

Note: Passing all the test cases on CodeCrunch does not guarantee that you will get full credit for the assignment. Self-testing is strongly encouraged.

Task A is a Greedy task and Task B is a Dynamic Programming task. Any inquiries regarding task A, please contact Le Quang Tuan (e0313526@u.nus.edu). Regarding task B, please contact Tran Tan Phat (e0196695@u.nus.edu)

Task A (4%)

Tuấn and Tít are two tomb raiders. Recently, they have discovered an ancient tomb of the Dragon King - the first king of Văn Lang (ancient Vietnam). There are N antique items in the tomb where N is an even number. Tuấn and Tít will decide to split those items so that Tuấn will get $\frac{N}{2}$ items and Tít will get $\frac{N}{2}$ items.

Since Tít graduated from law school and Tuấn graduated from computing school, they have different aesthetic sense. Formally, for the i-th item ($1 \leq i \leq N$), Tít gives it a_i aesthetic points and Tuấn gives it b_i aesthetic points. As law students are precise, Tít will not give two different items the same points, i.e all a_i are distinct.

Tít is too lazy to think. Therefore, she suggests the two to do the following process:

- Tuấn will repeatedly choose two items that haven't been chosen before, namely x and y ($x \neq y$).
- Tít will get item x if $a_x > a_y$, and get item y otherwise. Tuấn gets the remaining one.

The process is repeated $\frac{N}{2}$ times. Tuấn knows Tít well, so Tuấn knows all the a_i and all the b_i.

After this process, Tuấn gets a set S of items ($S \subset \{1, 2, ..., N\}$). Tuấn wants to maximize the total number of aesthetic points that he gives these items, i.e the sum $\sum_{x \in S} b_x$.

Although Tuấn have the right to decide the items to be chosen in each round, he does not know how to choose wisely (he got a D in CS3230). You, potentially having A+ in CS3230, might be able to help him.

In case you are stuck, hints will be given in the last page.

Input

The first line contains a positive integer N.
The next line contains N numbers, the i-th number contains one positive integer a_i.
The last line contains N numbers, the i-th number contains one positive integer b_i.

Output

A single line consists of the maximum points of the items Tuấn can get.

Limits

- $1 \leq N \leq 100000$

- $1 \leq a_i, b_i \leq 10^6$
- Your program should terminate within 1 second for C++11 and 2 seconds for Java
- Your algorithm should have a **worst case time complexity** of $O(N \log N)$.
- Your algorithm should use a **worst case memory complexity** of $O(N)$. Binary search trees, binary heaps, ... are all in $O(N)$ memory.

Scoring

Full credit (4 marks) will be given for solutions which run in **worst case time complexity** $O(N \log N)$. Solutions in **worst case time complexity** of $O(N^2)$ will be given a partial credit of 3 marks.

Example Input 1

```
6
14 10 11 18 5 6
1 7 6 12 15 1
```

Example Output 1

```
28
```

Explanation

The following table describes how Tuấn can achieve 28 points

Round	1	2	3
Tít	Item 6: 6 points	Item 1: 14 points	Item 4: 18 points
Tuấn	Item 5: 15 points	Item 3: 6 points	Item 2: 7 points

The cell in Tuấn's row contains the items Tuan gets and their aesthetic points given by him (b_x)
The cell in Tít's row contains the items Tít gets and their aesthetic points given by her(a_x)

Task B1 (4%)

Luna is the head receptionist of the Adventurer Guild at the town of Axel.

There are N adventurers in the Guild, each belongs to a class denoted by a positive integer. When there is an emergency alarm, all of them will rush to the Guild and form a line, waiting for further instructions. The i-th adventurer starting from the head of the line belongs to class $a[i]$.

Luna will then announce the battle plan, which divides the adventurers into parties such that these conditions hold:

- Each adventurer belongs to exactly one party.
- For ease of splitting, adventurers in a party should stand contiguously in the line.
- Each party should have adventurers from at least $minC$ and at most $maxC$ different classes. Otherwise, the party will not function properly.

Luna needs to calculate the number of possible battle plans modulo 10^9+7 for a report to the Guild. Two battle plans are considered different when there exists two adventurers in the same party in the first plan but in different parties in the second plan. Can you help her with the report?

Input

The first line contains 3 positive integers N, $minC$, $maxC$.
The second line contains N positive integers $a[i]$.

Output

A single number represents the number of battle plan modulo $10^9 + 7$.

Example Input

```
5 2 3
1 3 2 2 4
```

Example Output

```
2
```

Explanation

First way is to group first 2 adventurers into a party, which have 2 different classes (1 and 3) and last 3 adventurers into a different party, which have 2 different classes (2 and 4).
Second way is to group first 3 adventurers into a party, which have 3 different classes (1, 2 and 3) and last 2 adventurers into a different party, which have 2 different classes (2 and 4).

Limits

- $1 \leq N \leq 1000, 1 \leq minC \leq maxC \leq N, 1 \leq a[i] \leq 10^6$
- Your program should terminate within 1 second for C++11 and 2 seconds for Java
- Your algorithm should have a **worst case time complexity** of $O(N^2 + max(a[i]))$

Task B2 (Challenge question) (1%)

Recently, a huge number of new adventurers joined the Guild, inspired by a successful adventurer with the weakest class.
Can you help Luna with the report this time?

Input

The first line contains 3 positive integers N, $minC$, $maxC$.
The second line contains N positive integers $a[i]$.

Output

A single number represents the number of battle plan modulo $10^9 + 7$.

Limits

- $1 \leq N \leq 10^6, 1 \leq minC \leq maxC \leq N, 1 \leq a[i] \leq 10^6$
- Your program should terminate within 1 second for C++11 and 2 seconds for Java

Note

This question is graded on binary basis.

1 Hints for task A

Firstly, we can sort all the items so that $a_i < a_{i+1}$ for every $1 \leq i < N$. Then for every pair of items (i, j) where $i < j$ Tuan chooses, Tuan will get b_i points.

Now, consider the set of items Tuan gets (after sorted), namely $S \subset \{1, 2, ..., N\}$. The total number of points Tuan gets is $\sum_{x \in S} b_x$.

Because for each item i in S, Tuan has to pair it with another item j not in S such that $j > i$, this set S has a nice property on the bound (can be lower or upper) of the numbers of its items in each **prefix** of the items. Can you find out this property? A prefix at position i of the items is the array of items $\{1, 2, 3, .., i\}$.

For example, when $N = 4$, S can only be $\{1, 2\}$ or $\{1, 3\}$. S cannot be $\{2, 3\}$ as no element in S can be paired with 1.

When $N = 6$, S can only be $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 2, 5\}$, $\{1, 3, 4\}$, or $\{1, 3, 5\}$

Now, we can see the process of repeatedly choosing a pair (in the original problem) as a process of repeatedly choosing an element to put in S such that the property that we have just mentioned is preserved.

Lastly, find an order $id_1, id_2, ...id_N$ of items and then sequentially decide whether item id_i should be in S or not (by checking the above property) depending on the decision of all id_j $(j < i)$. Use exchange argument to prove that this decision leads to a global optimal solution.

This should give you a $O(n^2)$ algorithm.

It's kind of .. hard to get full marks. Use some data structure to quickly make a decision.

Another easier way to get full marks is trying to build up S from prefixes of items (from prefix at 1 to prefix at N) while maintaining the property of S. Delay to decide whenever possible, decide only when you need to push something into S to maintain its property.

The decision made should lead to an optimal solution, use exchange property to show that.

All the data structures learnt in CS2040 should be sufficient: Stack, Queue, Heap, AVL tree, ...

2 Hints for task B

Here is an example of a counting problems: https://www.geeksforgeeks.org/count-ways-reach-nth-stair-using-step-1-2-3/

Also, a reminder that **worst case** time complexity of insertion into HashMap is O(n), where n is number of existing elements in the structure.