Problem B. Proficient in I2A

Problem Description

A straightforward path to becoming proficient in "Introduction to Algorithms" is to become familiar with the STL (*Standard Template Library*).

Let's attempt to solve some simple tasks using the STL (although, it's entirely up to you whether or not to use it)!

Task 1

Given an integer array a of length n.

Output n numbers: the array a sorted from smallest to largest.

- Keywords: `std::sort`

Task 2

Given an integer array a of length n.

Output n numbers: the array a in reverse order.

- Keywords: `std::reverse`

Task 3

Given an integer array a of length n, while there are two adjacent elements that are the same, remove one of them.

Output the resulting array when there are no longer any adjacent elements that are the same.

- Keywords: `std::unique`

Task 4

Given an integer array a of length n, let array $c = (c[0], c[1], \ldots, c[n-1])$, where $c[i] = \sum_{k=0}^{i} a[k]$ be the prefix sum of a.

Output n numbers: the array c.

- Keywords: `std::partial_sum`

Task 5

Given an integer array a of length n.

Output 2 numbers: the 0-based index of the **first** minimum element and the **last** maximum element in the array a.

- Keywords: `std::min_element`, `std::max_element`

Task 6

Given an integer n.

Output all permutations of the first n lowercase Latin letters from lexicographically **largest** to **smallest**.

- Keywords: `std::next_permutation`, `std::prev_permutation`

Task 7

Given two **sorted** integer arrays a and b of length n, let array $c = (a[0], a[1], \ldots, a[n-1], b[0], b[1], \ldots, b[n-1])$ be the concatenation of a and b.

Output $2n$ numbers: the array c sorted from smallest to largest.

- Keywords: `std::merge`

Task 8

Given two integer arrays a and b of length n, process n queries.

We use $b[i]$ to denote the i^{th} query. Please output the smallest element in a that is larger than $b[i]$, output 0 if there is no element in a that is larger than $b[i]$.

- Keywords: `std::set`, `std::lower_bound`, `std::upper_bound`

Task 9

Given an integer array a of length n.

Output n numbers: the occurrences of $a[0], a[1], \ldots, a[n-1]$ in the array a.

- Keywords: `std::map`

Task 10

Process n queries to maintain a multiset (initially empty).

We use $a[i]$ to denote the i^{th} query. If $a[i] = 0$ and the multiset is non-empty, output and remove any smallest integer from the multiset, otherwise, insert $a[i]$ into the multiset.

- Keywords: `std::priority_queue`, `std::multiset`

Input Format

The first line of input contains a single integer T --- the number of test cases. The description of test cases follows.

The first line of each test case contains 2 integers *op*, n, denoting the task you should solve, and the length of the array (which is also the number of queries for $op \in \{8, 10\}$).

If $op \neq 6$, the second line of each test case contains n integers $a[0], a[1], \dots, a[n-1]$.

If $op \in \{7, 8\}$, the third line of each test case contains n integers $b[0], b[1], \dots, b[n-1]$.

Output Format

For each test case, output everything in a single line, with spaces in between.

Constraints

- $1 \leq T \leq 100$.
- $1 \leq op \leq 10$.
- $1 \leq n \leq 10\,000$.
- $n \leq 6$ if $op = 6$.
- $0 \leq a[i], b[i] \leq 100\,000$ for $i = 0, 1, \dots, n-1$.
- $a[0] \leq a[1] \leq \dots \leq a[n-1]$ and $b[0] \leq b[1] \leq \dots \leq b[n-1]$ if $op = 7$.
- All the inputs are integers.

Subtasks

1. (5 points) $op = 1$.
2. (5 points) $op = 2$.
3. (5 points) $op = 3$.
4. (5 points) $op = 4$.
5. (5 points) $op = 5$.
6. (5 points) $op = 6$.
7. (5 points) $op = 7$.
8. (15 points) $op = 8$.
9. (15 points) $op = 9$.
10. (15 points) $op = 10$.
11. (20 points) No additional constraints.

No.	Testdata Range	Time Limit (ms)	Memory Limit (KiB)
Samples	1	1000	262144
1	2	1000	262144
2	3	1000	262144
3	4	1000	262144
4	5	1000	262144
5	6	1000	262144
6	7	1000	262144
7	8	1000	262144
8	9	1000	262144
9	10	1000	262144
10	11	1000	262144
11	1-12	1000	262144

Samples

Sample Input 1

```
10
1 5
4 8 7 6 3
2 5
48 76 34 87 63
3 10
2 4 4 2 5 5 5 5 55 1
4 5
4 8 7 6 3
5 10
4 8 7 6 3 4 8 7 6 3
6 3
7 5
3 4 6 7 8
0 2 5 6 9
8 5
4 8 7 6 3
3 1 4 7 9
9 10
2 7 1 8 2 8 1 8 2 8
10 10
3 0 0 7 2 0 7 0 0 0
```

Sample Output 1

```
3 4 6 7 8
63 87 34 76 48
2 4 2 5 55 1
4 12 19 25 28
4 6
cba cab bca bac acb abc
0 2 3 4 5 6 6 7 8 9
4 3 6 8 0
3 1 2 4 3 4 2 4 3 4
3 0 2 7 7
```

References

- https://en.cppreference.com/w/
- https://cplusplus.com/