1.Even Number of Digits

Given an array of integers, find the elements which have an even number of digits.

Example

Array: [42, 564, 5775, 34, 123, 454, 1, 5, 45, 3556, 23442]

Answer: 42, 5775, 34, 45, 3556

The order of the returned elements should be the same as the order of the initial array.

Testing

Input Format

The first line contains 'T' denoting the no. of test cases.

Next T lines each contain a number 'n' denoting the number of elements, followed by n space-separated numbers denoting the array elements.

Output Format

T lines contain n numbers denoting the elements with even digits.

Sample Input

2

11 42 564 5775 34 123 454 1 5 45 3556 23442

3 11 4 200

Expected Output

42 5775 34 45 3556

11

Constraints

0 <= T <= 100

1 <= N <= 100

0 <= array element <= 10^6

```
class Solution {
        List<Integer> getEvenDigitNumbers (int[] arr) {
                // add your logic here
        }
}
```

2.Next Greater Permutation

Given an array, rearrange it to its next greater permutation. Do it in-place with extra constant memory only. Do not use any library function for the next permutation.

Example

Array: [1, 2, 3, 4]

Next Greater Permutation: [1, 2, 4, 3]

Next Greater Permutation: [1, 3, 2, 4]

Next Greater Permutation: [1, 3, 4, 2]

Next Greater Permutation: [1, 4, 2, 3]

Next Greater Permutation: [1, 4, 3, 2]

Next Greater Permutation: [2, 1, 3, 4]

If the next greater permutation does not exist, return the lowest possible order (sorted in ascending order).

Examples

Array: [4, 3, 2, 1]

Next Greater Permutation: [1, 2, 3, 4]

Array: [2, 2, 9]

Next Greater Permutation: [2, 9, 2]

Next Greater Permutation: [9, 2, 2]

Next Greater Permutation: [2, 2, 9]

Array: [4]

Next Greater Permutation: [4]

Testing

Input Format

The first line contains 'T' denoting the no. of test cases.

Next T lines each contain a number 'n' denoting the number of elements, followed by n space-separated numbers denoting the array elements.

Output Format

T lines each contain n numbers denoting the next greater permutation.

Sample Input

5

3 1 3 2

3 3 2 1

3 2 2 9

3 2 9 9

1 4

Expected Output

2 1 3

1 2 3

2 9 2

9 2 9

4

Constraints

0 <= T <= 100

1 <= N <= 10^5

1 <= value of array element <= 10^5

```
class Solution {
	void nextGreaterPermutation (int[] arr) {
		// add your logic here
	}
}
```

Inversion Count

The inversion count of an array denotes how far is the array from being sorted.

If the array is sorted, inversion count is 0. If the array is sorted in reverse order, the inversion count is maximum.

More formally, the inversion count of an array A is the number of pairs (i, j) such A[i] < A[j] and i > j.

Example

Lets take the following array:
8, 4, 1, 2

This array has an inversion count of 5.
(8, 4), (8, 1), (8, 2), (4, 1), (4, 2)

Given an array A, calculate the inversion count of the array.

Testing

Input Format

First-line contains 'T' denoting the number of test cases.

For each test case

- The first line contains an integer 'n' denoting the length of the array.
- The second line contains n space-separated integers of the array.

Output Format

One line for each test-case, denoting the inversion count of the array.

Sample Input

3

5

1 1 2 2 3

3

3 2 1

5

10 1 2 3 4

Expected Output

0

3

4

Constraints

$1 <= T <= 10$
$1 <= n <= 10^4$
$1 <= A_i <= 10^5$