(Sorting)

Problem-1

A **sentence** is a list of words that are separated by a single space with no leading or trailing spaces. Each word consists of lowercase and uppercase English letters.

A sentence can be **shuffled** by appending the **1-indexed word position** to each word then rearranging the words in the sentence.

- For example, the sentence `"This is a sentence"` can be shuffled as `"sentence4 a3 is2 This1"` or `"is2 sentence4 This1 a3"`.

Given a **shuffled sentence** `s` containing no more than `9` words, reconstruct and return *the original sentence*.

Example 1:

```
Input: s = "is2 sentence4 This1 a3"

Output: "This is a sentence"

Explanation: Sort the words in s to their original positions "This1 is2 a3
sentence4", then remove the numbers.
```

Example 2:

```
Input: s = "Myself2 Me1 I4 and3"

Output: "Me Myself and I"

Explanation: Sort the words in s to their original positions "Me1 Myself2 and3
I4", then remove the numbers.
```

Constraints:

- `2 <= s.length <= 200`
- `s` consists of lowercase and uppercase English letters, spaces, and digits from `1` to `9`.
- The number of words in `s` is between `1` and `9`.
- The words in `s` are separated by a single space.
- `s` contains no leading or trailing spaces.

Problem-2

You are given an array of strings `names`, and an array `heights` that consists of **distinct** positive integers. Both arrays are of length `n`.

For each index `i`, `names[i]` and `heights[i]` denote the name and height of the i^{th} person.

Return `names` *sorted in* ***descending*** *order by the people's heights*.

Example 1:

```
Input: names = ["Mary","John","Emma"], heights = [180,165,170]

Output: ["Mary","Emma","John"]

Explanation: Mary is the tallest, followed by Emma and John.
```

Example 2:

```
Input: names = ["Alice","Bob","Bob"], heights = [155,185,150]

Output: ["Bob","Alice","Bob"]

Explanation: The first Bob is the tallest, followed by Alice and the second Bob.
```

Constraints:

- `n == names.length == heights.length`
- `1 <= n <= 10^3`
- `1 <= names[i].length <= 20`
- `1 <= heights[i] <= 10^5`
- `names[i]` consists of lower and upper case English letters.
- All the values of `heights` are distinct.

Problem-3

There are `n` seats and `n` students in a room. You are given an array `seats` of length `n`, where `seats[i]` is the position of the `i`th seat. You are also given the array `students` of length `n`, where `students[j]` is the position of the `j`th student.

You may perform the following move any number of times:

- Increase or decrease the position of the `i`th student by `1` (i.e., moving the `i`th student from position `x` to `x + 1` or `x - 1`)

Return *the **minimum number of moves** required to move each student to a seat such that no two students are in the same seat.*

Note that there may be **multiple** seats or students in the **same** position at the beginning.

Example 1:

```
Input: seats = [3,1,5], students = [2,7,4]

Output: 4

Explanation: The students are moved as follows:

- The first student is moved from from position 2 to position 1 using 1 move.

- The second student is moved from from position 7 to position 5 using 2 moves.

- The third student is moved from from position 4 to position 3 using 1 move.

In total, 1 + 2 + 1 = 4 moves were used.
```

Example 2:

```
Input: seats = [4,1,5,9], students = [1,3,2,6]

Output: 7

Explanation: The students are moved as follows:

- The first student is not moved.

- The second student is moved from from position 3 to position 4 using 1 move.

- The third student is moved from from position 2 to position 5 using 3 moves.

- The fourth student is moved from from position 6 to position 9 using 3 moves.

In total, 0 + 1 + 3 + 3 = 7 moves were used.
```

Example 3:

Input: seats = [2,2,6,6], students = [1,3,2,6]

Output: 4

Explanation: Note that there are two seats at position 2 and two seats at position 6.

The students are moved as follows:

- The first student is moved from from position 1 to position 2 using 1 move.

- The second student is moved from from position 3 to position 6 using 3 moves.

- The third student is not moved.

- The fourth student is not moved.

In total, 1 + 3 + 0 + 0 = 4 moves were used.

Constraints:

- `n == seats.length == students.length`
- `1 <= n <= 100`
- `1 <= seats[i], students[j] <= 100`

Problem-4

The **product difference** between two pairs `(a, b)` and `(c, d)` is defined as `(a * b) - (c * d)`.

- For example, the product difference between `(5, 6)` and `(2, 7)` is `(5 * 6) - (2 * 7) = 16`.

Given an integer array `nums`, choose four **distinct** indices `w`, `x`, `y`, and `z` such that the **product difference** between pairs `(nums[w], nums[x])` and `(nums[y], nums[z])` is **maximized**.

Return *the **maximum** such product difference*.

Example 1:

```
Input: nums = [5,6,2,7,4]

Output: 34

Explanation: We can choose indices 1 and 3 for the first pair (6, 7) and indices 2 and 4 for the second pair (2, 4).

The product difference is (6 * 7) - (2 * 4) = 34.
```

Example 2:

```
Input: nums = [4,2,5,9,7,4,8]

Output: 64

Explanation: We can choose indices 3 and 6 for the first pair (9, 8) and indices 1 and 5 for the second pair (2, 4).

The product difference is (9 * 8) - (2 * 4) = 64.
```

Constraints:

- `4 <= nums.length <= 10^4`
- `1 <= nums[i] <= 10^4`

Problem-5

Given the array `nums`, for each `nums[i]` find out how many numbers in the array are smaller than it. That is, for each `nums[i]` you have to count the number of valid `j's` such that `j != i` **and** `nums[j] < nums[i]`.

Return the answer in an array.

Example 1:

```
Input: nums = [8,1,2,2,3]

Output: [4,0,1,1,3]

Explanation:

For nums[0]=8 there exist four smaller numbers than it (1, 2, 2 and 3).

For nums[1]=1 does not exist any smaller number than it.

For nums[2]=2 there exist one smaller number than it (1).

For nums[3]=2 there exist one smaller number than it (1).

For nums[4]=3 there exist three smaller numbers than it (1, 2 and 2).
```

Example 2:

```
Input: nums = [6,5,4,8]

Output: [2,1,0,3]
```

Example 3:

```
Input: nums = [7,7,7,7]

Output: [0,0,0,0]
```

Constraints:

- `2 <= nums.length <= 500`
- `0 <= nums[i] <= 100`