Problem-1 (Array)

Given an array `nums`. We define a running sum of an array as `runningSum[i] = sum(nums[0]…nums[i])`.

Return the running sum of `nums`.

Example 1:

```
Input: nums = [1,2,3,4]

Output: [1,3,6,10]

Explanation: Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].
```

Example 2:

```
Input: nums = [1,1,1,1,1]

Output: [1,2,3,4,5]

Explanation: Running sum is obtained as follows: [1, 1+1, 1+1+1, 1+1+1+1,
1+1+1+1+1].
```

Example 3:

```
Input: nums = [3,1,2,10,1]

Output: [3,4,6,16,17]
```

Constraints:

- `1 <= nums.length <= 1000`
- `-10^6 <= nums[i] <= 10^6`

Problem-2 (Sorting)

Given an integer array `nums` of `2n` integers, group these integers into `n` pairs `(a₁, b₁), (a₂, b₂), ..., (aₙ, bₙ)` such that the sum of `min(aᵢ, bᵢ)` for all `i` is **maximized**. Return *the maximized sum*.

Example 1:

```
Input: nums = [1,4,3,2]

Output: 4

Explanation: All possible pairings (ignoring the ordering of elements) are:

1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3

2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3

3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4

So the maximum possible sum is 4.
```

Example 2:

```
Input: nums = [6,2,6,5,1,2]

Output: 9

Explanation: The optimal pairing is (2, 1), (2, 5), (6, 6). min(2, 1) + min(2, 5)
+ min(6, 6) = 1 + 2 + 6 = 9.
```

Problem-3 (Array)

Given two string arrays `word1` and `word2`, return `true` *if the two arrays* ***represent*** *the same string, and* `false` *otherwise.*

A string is **represented** by an array if the array elements concatenated **in order** forms the string.

Example 1:

```
Input: word1 = ["ab", "c"], word2 = ["a", "bc"]

Output: true

Explanation:

word1 represents string "ab" + "c" -> "abc"

word2 represents string "a" + "bc" -> "abc"

The strings are the same, so return true.
```

Example 2:

```
Input: word1 = ["a", "cb"], word2 = ["ab", "c"]

Output: false
```

Example 3:

```
Input: word1  = ["abc", "d", "defg"], word2 = ["abcddefg"]

Output: true
```

Constraints:

- `1 <= word1.length, word2.length <= 10^3`
- `1 <= word1[i].length, word2[i].length <= 10^3`
- `1 <= sum(word1[i].length), sum(word2[i].length) <= 10^3`
- `word1[i]` and `word2[i]` consist of lowercase letters.

Problem-4 (Sorting)

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`