Sorting: Bubble Sort ☆

Problem Submissions Leaderboard Editorial 🔒

Check out the resources on the page's right side to learn more about bubble sort. The video tutorial is by Gayle Laakmann McDowell, author of the best-selling interview book Cracking the Coding Interview.

Consider the following version of Bubble Sort:

```
for (int i = 0; i < n; i++) {

    for (int j = 0; j < n - 1; j++) {
        // Swap adjacent elements if they are in decreasing order
        if (a[j] > a[j + 1]) {
            swap(a[j], a[j + 1]);
        }
    }

}
```

Given an array of integers, sort the array in ascending order using the *Bubble Sort* algorithm above. Once sorted, print the following three lines:

1. `Array is sorted in numSwaps swaps.`, where $numSwaps$ is the number of swaps that took place.
2. `First Element: firstElement`, where $firstElement$ is the *first* element in the sorted array.
3. `Last Element: lastElement`, where $lastElement$ is the *last* element in the sorted array.

Hint: To complete this challenge, you must add a variable that keeps a running tally of *all* swaps that occur during execution.

For example, given a worst-case but small array to sort: $a = [6, 4, 1]$ we go through the following steps:

```
swap    a
0       [6,4,1]
1       [4,6,1]
2       [4,1,6]
3       [1,4,6]
```

It took 3 swaps to sort the array. Output would be

```
Array is sorted in 3 swaps.
First Element: 1
Last Element: 6
```

Function Description

Complete the function *countSwaps* in the editor below. It should print the three lines required, then return.

countSwaps has the following parameter(s):

- *a*: an array of integers .

Input Format

The first line contains an integer, n, the size of the array a.

The second line contains n space-separated integers $a[i]$.

Constraints

- $2 \leq n \leq 600$
- $1 \leq a[i] \leq 2 \times 10^6$

Output Format

You must print the following three lines of output:

1. `Array is sorted in numSwaps swaps.`, where $numSwaps$ is the number of swaps that took place.
2. `First Element: firstElement`, where $firstElement$ is the *first* element in the sorted array.
3. `Last Element: lastElement`, where $lastElement$ is the *last* element in the sorted array.

Sample Input 0

```
3
1 2 3
```

Sample Output 0

```
Array is sorted in 0 swaps.
First Element: 1
Last Element: 3
```

Explanation 0

The array is already sorted, so 0 swaps take place and we print the necessary three lines of output shown above.

Sample Input 1

```
3
3 2 1
```

Sample Output 1

```
Array is sorted in 3 swaps.
First Element: 1
Last Element: 3
```

Explanation 1

The array is *not sorted*, and its initial values are: $\{3, 2, 1\}$. The following 3 swaps take place:

1. $\{3, 2, 1\} \rightarrow \{2, 3, 1\}$
2. $\{2, 3, 1\} \rightarrow \{2, 1, 3\}$
3. $\{2, 1, 3\} \rightarrow \{1, 2, 3\}$

At this point the array is sorted and we print the necessary three lines of output shown above.