Implement Bubble Sort

This is the first of several challenges on sorting algorithms. Given an array of unsorted items, we want to be able to return a sorted array. We will see several different methods to do this and learn some tradeoffs between these different approaches. While most modern languages have built-in sorting methods for operations like this, it is still important to understand some of the common basic approaches and learn how they can be implemented.

Here we will see bubble sort. The bubble sort method starts at the beginning of an unsorted array and 'bubbles up' unsorted values towards the end, iterating through the array until it is completely sorted. It does this by comparing adjacent items and swapping them if they are out of order. The method continues looping through the array until no swaps occur at which point the array is sorted.

This method requires multiple iterations through the array and for average and worst cases has quadratic time complexity. While simple, it is usually impractical in most situations.

Instructions: Write a function `bubbleSort` which takes an array of integers as input and returns an array of these integers in sorted order from least to greatest.

⚗	`bubbleSort` should be a function.
⚗	`bubbleSort` should return a sorted array (least to greatest).
⚗	`bubbleSort([1,4,2,8,345,123,43,32,5643,63,123,43,2,55,1,234,92])` should return an array that is unchanged except for order.
⚗	`bubbleSort` should not use the built-in `.sort()` method.