Implement Merge Sort

Another common intermediate sorting algorithm is merge sort. Like quick sort, merge sort also uses a divide-and-conquer, recursive methodology to sort an array. It takes advantage of the fact that it is relatively easy to sort two arrays as long as each is sorted in the first place. But we'll start with only one array as input, so how do we get to two sorted arrays from that? Well, we can recursively divide the original input in two until we reach the base case of an array with one item. A single-item array is naturally sorted, so then we can start combining. This combination will unwind the recursive calls that split the original array, eventually producing a final sorted array of all the elements. The steps of merge sort, then, are:

1) Recursively split the input array in half until a sub-array with only one element is produced.

2) Merge each sorted sub-array together to produce the final sorted array.

Merge sort is an efficient sorting method, with time complexity of *O(nlog(n))*. This algorithm is popular because it is performant and relatively easy to implement.

As an aside, this will be the last sorting algorithm we cover here. However, later in the section on tree data structures we will describe heap sort, another efficient sorting method that requires a binary heap in its implementation.

Instructions: Write a function `mergeSort` which takes an array of integers as input and returns an array of these integers in sorted order from least to greatest. A good way to implement this is to write one function, for instance `merge`, which is responsible for merging two sorted arrays, and another function, for instance `mergeSort`, which is responsible for the recursion that produces single-item arrays to feed into merge. Good luck!

⚗	`mergeSort` should be a function.
⚗	`mergeSort` should return a sorted array (least to greatest).

🧪	`mergeSort([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.
🧪	`mergeSort` should not use the built-in `.sort()` method.