Searching/Sorting

What To Do:

Follow each step carefully. After finishing, submit your code to the autograder.

Questions

1. Recall the sorting algorithms from lecture. Those sorting algorithms were all comparison-based, which means that they compare elements to determine the sorting order. Contrastingly, in this lab you will implement a non-comparison-based sorting algorithm called the *radix sort.*

   All of the code you will write should be inside a class called `RadixSort`. Your tests should be inside a class called `RadixSortTest`. In this class, you should exhaustively test all of the below methods, and your grade will primarily come from the exhaustiveness and detail of your tests.

   (a) First, design the `static Optional<Integer> findMaxDigits(List<Integer> ls)` method that, when given a `List<Integer>` such that each element is strictly greater than 0, returns the number of digits that are in the largest element. If the given list contains no elements, return an empty `Optional`.

   (b) Second, design the `static List<Integer> combineBuckets(List<List<Integer>> buckets)` method that returns a new `List<Integer>` by combining all of the given elements in the list of lists. For example, `[[1, 2], [3], [4, 5]]` returns `[1, 2, 3, 4, 5]`.

   (c) Third, design the `static List<List<Integer>> createBuckets(List<Integer> ls, int digitIdx)` method that returns a list of "buckets," where each bucket denotes a digit from 0-9. The purpose of this method is to sort the values in `ls` by the digit value of `digitIdx`.

   For example, if the given list is `[42, 81, 2, 3, 771, 94]`, and `digitIdx` is 0, then we will sort the numbers based on the one's digit. So, we create ten buckets, one for each digit, then place each number into one of the buckets. This example returns the following list of buckets: `[[], [81, 771], [42, 2], [3], [94], [], [], [], [], []]`. Then, we combine them into one list, and sort based on the ten's place. Repeat this process until there are no more digits.

   (d) Finally, design the `static List<Integer> radixSort(List<Integer> ls)` method that, when given a `List<Integer>` such that each element is strictly greater than 0, sorts the elements using the radix sort algorithm. The idea of the algorithm, as suggested by the above steps, is to sort based on each place value, starting from the ones and up as far as needed. Your algorithm will require the methods that you designed in parts (a), (b), and (c), so make sure they are fully tested and work prior to starting this part.