Tips for HW05

- Tips for Recursion:
    - We use recursive methods in order to break larger problems down into many smaller problems, and we return back up the chain of method calls once we've found our "base case."
    - When designing recursive methods, considering your base case may be a good way to start. Ask yourself - "What input would I need to give the method to do the computations in one step?" For example - with a factorial recursive method, our base case would be calculating 0!, since we know for a fact that 0! = 1.
    - From there, we want to figure out a way to lead our initial problem to our base case with each recursive call.

- mergeSort():

    - You don't have to implement mergeSort from scratch, since RecursionUtils.java has helper methods to do some of the heavy lifting for you.

    - copyOfRange():
        - This should be used **before** your recursive calls in mergeSort.
        - This can be used to split your unsorted array into the left and right arrays.

    - merge():
        - This should be used **after** your recursive calls in mergeSort.
        - Write two String arrays, and show the output.

- mergeAll():
    - Consider merging two of the string arrays on each call of mergeAll.
    - The base case could occur when there's only one String array in the String[][] parameter.

- countDuplicates():
    - Consider counting the number of **one** unique element in the array on each iteration.
    - copyOfRange() could be useful here in order to shorten the array on each recursive call.

- verifyPalindrome():
    - Consider looking at the front and back letter and removing them from the String using String.substring() on each recursive call.

- numInteriorPoints():
    - Look at Point.java for information on the Point object.
    - Consider determining whether one Point is within the given radius on each recursive call and sum the number of valid points in the return statements.