COW - Arrays

Level 1

Complete the following methods in the ArrayPrinter Class.
Name: print
Input: int [] array, Intake feed
Output: nothing
Action: passes into feed.give() all the elements of the array using a loop.

Complete the following methods in the ArrayCounter Class.
Name: countPositives
Input: int [] numbers
Output: int countPositives
Action: returns the number of positive numbers stored in the array

Complete the following methods in the ArrayModifier Class.
Name: flip
Input: double [] data
Output: double [] flippedData
Action: Creates a new array the same size as the old array. Each element of the new array should be equal to the negated value of the corresponding element in data.
So {-10, -3, 1, 2, 6, 2, -6} becomes {10, 3, -1, -2, -6, -2, 6}

Complete the following methods in the StatisticalCalculator Class.
Name: sumArray
Input: double [] values
Output: double sum
Action: calculates the sum of all the elements in the array.

Complete the following methods in the ArrayAnalyzer Class.
Name: hasValue
Input: int[] array, int value
Output: boolean hasValue
Action: returns whether the array is storing the value passed in

Level 2

Complete the following methods in the ArrayPrinter Class.

Name: printReverse
Input: int [] array, Intake feed
Output: nothing
Action: passes into feed.give() all the elements of the array in reverse.

Complete the following methods in the ArrayCounter Class.

Name: countNegativeOdds
Input: int [] array
Output: int countNegativeOdds
Action: returns the number of negative odds numbers stored in the array

Complete the following methods in the ArrayModifier Class.

Name: amplify
Input: double [] data, double multiplier
Output: double [] amplifiedData
Action: Creates a new array the same size as the old array. Each element of the new array should be equal to the corresponding element in data multiplied by multiplier.
So {-10, -3, 1, 2, 6, 2, -6}, 2 becomes {-20, -6, 2, 4, 12, 4, -12}

Complete the following methods in the StatisticalCalculator Class.

Name: getAverageValue
Input: double [] values
Output: double average
Action: calculates the average of all the elements in the array.

Complete the following methods in the ArrayAnalyzer Class.

Name: allTheSame
Input: int[] array
Output: boolean allTheSame
Action: returns whether all the values in the array are the exact same. You can assume that the array has at least one value

Level 3

Complete the following methods in the ArrayPrinter Class.

Name: printEveryOtherElement
Input: int [] array, Intake feed
Output: nothing
Action: passes into feed.give() every other elements of the array in starting with the first one.

Complete the following methods in the ArrayCounter Class.

Name: countInRange
Input: int [] numbers, int min, int max
Output: int countInRange
Action: returns the number of elements in the array that are between min and max inclusive.

Complete the following methods in the ArrayModifier Class.

Name: cap
Input: double [] data, int min, int max
Output: double [] cappedData
Action: Creates a new array the same size as the old array. Each element of the new array should be equal to the corresponding element in data unless it is less then min or greater than max. If it falls out of range then it should be set to min or max depending on whether it was too low or too high.
So {-10, -3, 1, 2, 6, 2, -6}, -5, 5 becomes {-5, -3, 1, 2, 5, 2, -5}

Complete the following methods in the StatisticalCalculator Class.

Name: getMedianValue
Input: double [] values
Output: double median
Action: returns the median value of the array passed in. You can assume that the array passed in is sorted!!!! Note that for even numbers arrays the median is the average of the middle two values.

Complete the following methods in the ArrayAnalyzer Class.

Name: isDecending
Input: int[] array
Output: boolean isDecending
Action: returns true if all the numbers are descending. If two sequential numbers are the exact same, then that is not considered descending.

Level 4

Complete the following methods in the ArrayPrinter Class.

Name: printFirstHalf
Input: int [] array, Intake feed
Output: nothing
Action: passes into feed.give() all the values in the first half of the array not including the middle value of odd numbered arrays.

Complete the following methods in the ArrayCounter Class.

Name: countPairs
Input: int [] numbers
Output: int countOfPairs
Action: returns the number of elements in the array that match either the element in front of it or the element in back of it. Hint – count the end values as special cases and loop through the rest.

Complete the following methods in the ArrayModifier Class.

Name: averageElements
Input: double [] data1, double [] data2
Output: double [] averagedData
Action: Creates a new array the same size as data1 and data2 and stores the average of the corresponding values. You may assume that arrays data1 and data2 have the same length. So {-10, -3, 1, 2, 6, 2, -6}, {10, -7, 4, 3, 6, 0, -8} becomes {0, -5, 2.5, 2.5, 6, 1, -7},

Complete the following methods in the StatisticalCalculator Class.

Name: standardDeviation
Input: double [] values
Output: double deviation
Action: calculates the standard deviation of the numbers in values

Complete the following methods in the ArrayAnalyzer Class.

Name: equal
Input: String [] arr1, String [] arr2
Output: boolean equal
Action: returns whether all the two arrays are the same. For two arrays to be the exact same, they must have the same size and each set of corresponding values have to be the same.

Level 5

Complete the following methods in the ArrayPrinter Class.

Name: printSecondHalf
Input: int [] array, Intake feed
Output: nothing
Action: passes into feed.give() all the values in the second half of the array not including the middle value of odd numbered arrays.

Complete the following methods in the ArrayCounter Class.

Name: countUniqueElements
Input: int [] numbers
Output: int countOfUniqueElements
Action: returns how many elements in the array are unique. You can assume the elements are in order. So {1, 1, 2, 3, 3, 4, 5, 5, 5, 6} returns 3

Complete the following methods in the ArrayModifier Class.

Name: evenOut
Input: double [] data
Output: double [] evenedOutData
Action: Creates a new array the same size as the old array. Each element of the new array should be the average of the corresponding element in data and the two adjacent elements in data. The two elements at the end of evenedOutData should be an average of the last two elements in data.
So {-10, -3, 1, 2, 6, 2, -6} becomes {-6.5, -4, 0, 3, 3.333333, 0.6666666, -2}

Complete the following methods in the StatisticalCalculator Class.

Name: getDeviations
Input: double [] values
Output: double [] zScores
Action: returns an array the same size as values that stores the corresponding z-score for each value in values

Complete the following methods in the ArrayAnalyzer Class.

Name: inOrder
Input: int[] array
Output: boolean isOrder
Action: returns true if all the numbers are either all descending or ascending.

Level 6

Complete the following methods in the ArrayPrinter Class.

Name: printSection
Input: int [] array, Intake feed, int startIndex, int endIndex
Output: nothing
Action: passes into feed.give() all the values between startIndex and endIndex inclusive. startIndex and endIndex might or might not be in order. The order of elements should progress from start to end even if that mean they are in reverse order. There is also the possibility that they are out of bounds. If that is the case then make sure that only in bound elements are referenced and feed into feed.give().

Complete the following methods in the ArrayCounter Class.

Name: countNonUniqueElements
Input: int [] numbers
Output: int countOfNonUnique
Action: returns how many numbers show up multiple times in the array. It should only do it once for each value. You can assume the elements are in order. So {1, 1, 2, 3, 3, 4, 5, 5, 5, 6} returns 3.

Complete the following methods in the ArrayModifier Class.

Name: compress
Input: double [] data
Output: double [] compressedData
Action: Creates a new array the half the size as the old array. Every sequential pair of elements in the old array should be averaged together to become one element in the new array. If the old array has an odd number of elements and thus has an element at the end without a corresponding pair, then it is averaged in with the previous two.
So {-10, -3, 1, 2, 6, 2, -6} becomes {-6.5, 1.5, 2}

Complete the following methods in the StatisticalCalculator Class.

Name: getCorrelation
Input: double [] xValues, double [] yValues
Output: double correlation
Action: calculates the correlation for the sets of corresponding x and y values

Complete the following methods in the ArrayAnalyzer Class.

Name: hasTwoUniquePairs
Input: int[] array
Output: boolean hasTwoUniquePairs
Action: returns true if the array has two unique pairs of sequential matching numbers. This means that the two pairs do not share any values. Ex: {2, 2, 6, 4, 7, 7, 5}