List COW

Level 1

Complete the following methods in the ListPractice1 Class.

Name: print
Input: ArrayList<Integer> numbers, Intake feed
Output: nothing
Action: passes into feed.give() all the elements of the list using a loop.

Name: printReverse
Input: ArrayList<Integer> numbers, Intake feed
Output: nothing
Action: passes into feed.give() all the elements of the list in reverse.

Name: printPositives
Input: ArrayList<Integer> numbers, Intake feed
Output: nothing
Action: passes into feed.give() all the positive numbers in the list.

Name: printNegativeOdds
Input: ArrayList<Integer> numbers, Intake feed
Output: nothing
Action: passes into feed.give() all the negative odd numbers in the list.

Name: printMultiples
Input: ArrayList<Integer> numbers, Intake feed, int num
Output: nothing
Action: passes into feed.give() all the numbers that are a multiple of num

Level 2

Complete the following methods in the ListPractice2 Class.

Name: generateReverse
Input: ArrayList<Integer> numbers
Output: ArrayList<Integer> reversed
Action: Creates and returns an ArrayList that stores everything that numbers stores but in reverse order.

Name: generatePositiveEvens
Input: ArrayList<Integer> numbers
Output: ArrayList<Integer> positiveEvens
Action: Creates and returns an ArrayList that stores all the positive even Integers stored in numbers

Name: generateInRange
Input: ArrayList<Integer> numbers, int min, int max
Output: ArrayList<Integer> inRange
Action: Creates and returns an ArrayList that stores all the Integers stored in numbers that are between min and max (inclusive).

Name: generateFirstMiddleAndLast
Input: ArrayList<Integer> numbers
Output: ArrayList<Integer> firstMiddleLast
Action: Creates and returns an ArrayList that stores all the first, middle, and last Integers stored in numbers. If the list has an even number of elements, then it stores the average of the middle two elements. You may assume that the list has at least one element.

Name: generateFirstHalf
Input: ArrayList<Integer> numbers
Output: ArrayList<Integer> firstHalf
Action: Creates and returns an ArrayList that stores the first half of the elements in numbers not including the middle value of lists with an odd number of values

Level 3

Complete ListPractice3 Class.

Name: print
Input: ArrayList<String> words
Output: nothing
Action: print out all the elements of the array

Name: combine
Input: ArrayList<String> words1, ArrayList<String> words2
Output: ArrayList<String> combined
Action: combines the elements in the first list with the elements in the second list into one larger list. All the elements in words1 come first followed by the elements in words2

Name: subArray
Input: ArrayList<String> words, int i1, int i2
Output: ArrayList<String> subList
Action: returns a list that only includes the elements from the first index up to and including the element at the second index.

Name: equal
Input: ArrayList<String> words1, ArrayList<String> words2
Output: boolean areEqual
Action: returns whether the two lists are exactly the same.

Name: contains
Input: ArrayList<String> words, String word
Output: boolean isThere
Action: returns whether or not word is contained within the list

Level 4

For this level, you will be using the ColorStrip Class. The Color Strip has the following methods:
getRed()
getGreen()
getBlue()

Complete the following methods in the ColorCoder Class. Note that there is a class variable that is an ArrayList of ColorStrip's called *stripes*.

Name: findBrightest
Input: nothing
Output: int indexOfBrightest
Action: returns the index of where the brightest Color Strip is located in the list stripes. The brightness is determined by adding the red, green, and blue values.

Name: findAverage
Input: nothing
Output: ColorStrip averageStrip
Action: generates a ColorStrip that has an average of all of the color values of the ColorStrips in the list stripes.

Name: shiftLeft
Input: nothing
Output: nothing
Action: removes the first strip in the list stripes and places it at the end.

Name: areUniform
Input: ColorStrip strip1, ColorStrip strip2
Output: boolean areUniform
Action: determines if the two strips passed in have the same dominant color. The dominant color is the color value that is higher than the other two.

Name: isUniform
Input: nothing
Output: boolean isUniform
Action: determines if all the stripes in the list have the same dominant color. The dominant color is the color value that is higher than the other two.

Level 5

Complete the following methods in the Editor Class:

Name: findFirst
Input: ArrayList<String> words, String wordToFind
Output: int index
Action: returns the index of where the wordToFind first shows up in words. If the word is not found then a -1 is returned. For example, findFirst({“ape”, “bat”, “cat”, “ape”, “ape”, “bat”, “ape”}, “bat”) returns 1

Name: findLast
Input: ArrayList<String> words, String wordToFind
Output: int index
Action: returns the index of where the wordToFind last shows up in words. If the word is not found then a -1 is returned. For example, findLast({“ape”, “bat”, “cat”, “ape”, “ape”, “bat”, “ape”}, “bat”) returns 5

Name: remove
Input: ArrayList<String> words, String wordToRemove
Output: ArrayList<String> words
Action: removes all instances of wordToRemove from words and returns the list. For example, remove({“ape”, “bat”, “cat”, “ape”, “ape”, “bat”, “ape”}, “ape”) returns {“bat”, “cat”, “bat”}

Name: remove
Input: ArrayList<String> words, ArrayList<String> wordsToRemove
Output: ArrayList<String> words
Action: removes from words every instance of any of the words stored in wordsToRemove. For example, remove({“ape”, “bat”, “cat”, “ape”, “ape”, “bat”, “ape”}, {“ape”, “cat”, “dog”}) returns {“bat”, “bat”}

Name: replace
Input: ArrayList<String> words, String wordToRemove, String wordToInsert
Output: ArrayList<String> words
Action: replaces all instances of wordToRemove in words with wordToInsert and returns the list. For example, remove({“ape”, “bat”, “cat”, “ape”, “ape”, “bat”, “ape”}, “ape”, “dog”) returns {“dog”, “bat”, “cat”, “dog”, “dog”, “bat”, “dog”}

Level 6

Complete the following methods in the Analyzer Class:

Name: getOccurences
Input: ArrayList<String> words, String wordToFind
Output: int count
Action: counts the number of instances of wordToFind in words. For example, getOccurences({“ape”, “bat”, “cat”, “ape”, “ape”, “bat”, “ape”}, “ape”) returns 4

Name: countRepititions
Input: ArrayList<String> words
Output: int count
Action: counts the number of words in words that show up more than once. For example, countRepitions({“ape”, “bat”, “cat”, “ape”, “ape”, “bat”, “ape”}) returns 2. Hint – you might need to remove words to avoid double counting.

Name: getNumberInCommon
Input: ArrayList<String> words1, ArrayList<String> words2
Output: int count
Action: returns the number of words the two lists have in common. You may assume that there are no repetitions but not in any order. For example, getNumberInCommon({“ape”, “bat”, “cat”, “dog”}, {“ape”, “dog”, “eel”, “fox”}) returns 2

Name: getNumberDifferent
Input: ArrayList<String> words1, ArrayList<String> words2
Output: int count
Action: returns the number of words that show up in one of the lists but not the other. You may assume that there are no repetitions but not in any order. For example, getNumberDifferent({“ape”, “bat”, “cat”, “dog”}, {“ape”, “dog”, “eel”, “fox”}) returns 4. Hint – there is a shortcut to do this.

Name: getMostCommon
Input: ArrayList<String> words
Output: String word
Action: returns the word that show up the most in words. If two or more words tie for showing up the most frequently, then the method returns any of the words that show up the most frequently. For example, getMostCommon({“ape”, “bat”, “cat”, “ape”, “ape”, “bat”, “ape”}) returns “ape” and getMostCommon({“ape”, “bat”, “cat”, “bat”, “ape”}) returns “ape” or “bat”