CS 245 - Lab Assignment 4 Fall, 2020

Lab Assignment 4 - Merge K Sorted Lists

The goal of this assignment is to demonstrate your mastery of sorting algorithms by merging an arbitrary number of arrays of sorted values.

Background

Given an "*outer*" array of *k* "*inner*" arrays, with each inner array sorted in ascending order. Your task is to merge and return all the inner arrays into one sorted "*output*" array.

Examples

Below, find four examples of the inputs (outer array) and the corresponding output array.

Example 1
Input: outerArray = `[ [1.1, 4,4, 5,5], [1.1, 3.3, 4.4], [2.2, 6.6] ]`
Output: `[1.1, 1.1, 2.2, 3.3, 4.4, 4.4, 5.5, 6.6]`

Example 2
Input: outerArray = `[]`
Output: `[]`

Example 3
Input: outerArray = `[[]]`
Output: `[]`

Example 4
Input: outerArray = `[[9.7, 17.1], [15.8], [12.7, 18.5, 21.27]]`
Output: `[9.7, 12.7, 15.8, 17.1, 18.5, 21.27]`

Requirements

Requirement 1: "KLists" class
Your implementation must be encapsulated in a Java class called "KLists". The entire implementation must be contained within this class. The file in which this class is written must be named KLists.java.

Requirement 2: mergeKLists function

Your implementation must contain one public function with the following header:

```
public double [] mergeKLists (double [][] outerArray)
```

Submission

In a unique repository on github, check all source code. If you are unsure about how to create a (free) Github account, speak with the instructor or either of the TAs. The source code must be in Java. If your github repository is not public, you must allow access for the following user IDs: dbrizan, dennisdang17, huntermrocha. Place the URL for this repository on Canvas. You may also add any comments in a README file (text, PDF or Word document) to help the grader understand or execute your implementation.

Grading

Your grade for this assignment will be determined as follows:

- 75% = Implementation: your class implementations must run successfully with the source files and data provided. It must produce the expected results, a sample of which appears in the Implementation section above. Any deviation from the expected results results in 0 credit for implementation.
- 15% = Decomposition: in the eyes of the grader, your solution follows the suggestions above or otherwise must represent a reasonable object-oriented and procedural decomposition to this problem.
- 10% = Style: your code must be readable to the point of being self-documenting; in other words, it must have consistent comments describing the purpose of each class and the purpose of each function within a class. Names for variables and functions must be descriptive, and any code which is not straightforward or is in any way difficult to understand must be described with comments.