Quiz #2 (10 marks)

On CourseWeb: Course Documents ↪ Week 11 Materials ↪ Quiz 2

Submission Instructions: Submit your code as a single zip file at the CourseWeb page of the quiz. Name your zip file `<Pitt username>.zip` (e.g., `ksm73.zip`). Remember to include all source-code files including the ones supplied to you. Make sure that your zip file doesn't contain any folders. **You only have one submission attempt.**

General Guidelines

- Build your solution incrementally. On short intervals, make sure that the following commands produce no compilation errors. `javac *.java; java Quiz2Test`
- Do not change anything in all supplied files except `Quiz2.java`.
- Before submission, make sure that the following commands produce no error and give the output that all tests are passed. In grading your submissions, a couple new test cases will be used as well.

  ```
  javac *.java; java Quiz2Test
  ```

Question

1. Download the following files from the Quiz page on CourseWeb.
   `ListInterface.java, AList.java, LList.java, Quiz2Test.java, Quiz2.java`.
2. Complete the implementation of the following two methods inside the file Quiz2.java.

   -
     ```
     public static <T extends Comparable<? super T>>
             int countInversions(ListInterface<T> list) {...}
     ```

     The method counts the number of inversions in a list of Comparable items. Consider every pair of values in the list. There are n(n–1)/2 pairs. Each pair that is out of order contributes one to the number of inversions. For example, consider the list [5, 6, 1, 5]. It has 6 pairs: (5,6), (5,1), (5,5), (6,1), (6,5), and (1,5). The pairs (5,1), (6,1), and (6,5) are out of order and should be counted as inversions. So, the number of inversions = 3. A sorted list has zero inversions. A reverse sorted list has n(n–1)/2 inversions. Don't forget that list indexes start from 1.

   -
     ```
     public static <T extends Comparable<? super T>>
             int shuffle(ListInterface<T> list,
                         ListInterface<Integer> pairs) {...}
     ```

     This method takes a list of Comparable items (`list`) and a list of pairs of positions (`pairs`). A pair is two consecutive entries in the `pairs` list. `pairs` entries number 1 and 2 form the first pair, entries 2 and 3 the second, entries 3 and 4 the third pair, and so on. For example, consider the `pairs` list [3, 7, 1, 5]. It has 3 pairs: (3,7), (7,1), and (1,5). For each pair of positions, the method checks the list entries located at these positions. It swaps the items if they are out of order. The method returns the number of inversions after all the pairs have been processed. Note that the pair elements are not necessarily in order.

Grading Criteria

All test cases are passed including the unseen ones	10
Some unseen test cases are passed	9
No unseen test cases are passed	5-7
Some seen test cases are not passed	6
Code doesn't compile	0-5