Lab 5: Permutation using Recursion

```
Location: CourseWeb -> Labs/Recitations -> Lab 5: Permutation
Download the following files:
  1. Permutation.java
  2. PermutationTester.java
```

Introduction

In mathematics, the notion of **permutation** relates to the act of rearranging, or permuting, all the members of a list into some sequence or order. For example, there are six permutations of the list `[1,2,3]`, namely: `[1,2,3]`, `[1,3,2]`, `[2,1,3]`, `[2,3,1]`, `[3,1,2]`, and `[3,2,1]`. The number of permutations of n distinct objects is n factorial ($n!$).

What to do

Implement the **recursive** method named `permutation` (marked `TO DO`) which can be found in `Permutation.java`. This method takes an `ArrayList<Integer>` as an argument and returns `ArrayList<ArrayList<Integer>>` as a result. For example, suppose you have a list of class wrapper `Integer` named `list` which contains `[1,2,3]` (from the first index to the last index). By calling the method `permutation` using the following statement (`Permutation` is a static class):

```
ArrayList<ArrayList<Integer>> result = Permutation.permutation(list);
```

The result should be `[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]` (not necessary in that order). To generate this, you can perform the following steps:

1. Remove the first index of the list and store it in a variable (let's named this variable `firstEntry`)
2. Permute the rest of the list which should give you a list of list (let's call this `result`)
3. For every list inside `result`, insert `firstEntry` into every valid index. Every time you insert, you get a new list which will be a member of your final result.

For example, start with the list `[1,2,3]`, at step 1, you store 1 in `firstEntry` and the rest of the list is `[2,3]`. After you permute the list `[2,3]`, you should get the list of list `[[2,3],[3,2]]` (let's call this `result`. The first index of `result` is the list `[2,3]`. In that list, there are three valid indexes, 0, 1, and 2, that we can insert an item. By inserting `firstEntry` into `[2,3]` at all possible valid indexes, you will get three new lists, `[1,2,3]`, `[2,1,3]`, and `[2,3,1]`. The second index of the `result` is the list `[3,2]`. Again, by inserting `firstEntry` into `[3,2]` at all possible valid indexes, you will get three new lists, `[1,3,2]`, `[3,1,2]`, and `[3,2,1]`. Note that all new lists are permutation of the list `[1,2,3]`. You can see that the step 2 above is where recursive call happens. Do not forget about the base case. Think about the base case by yourselves.

Note that it may be a good idea to create your own test class before running the given test class. To see the content of an `ArrayList<Integer>` named `aList`, you can simply use the statement `System.out.println(aList)`.

Test Class

A test class (`PermutationTester.java`) is provided. Simply compile and run this test class. This program will test your `Permutation.java` and show your total points (out of 10). If you do not get the full 10 points, you should keep trying to fix your code until you get 10 points. **Note** that this test class is not perfect. It cannot tell you why your program is incorrect. You may have to look at the source code of `PermutationTester.java` and see why it says `FAIL` and trace your code.

Due Date and Submission

For the due date, please check the lab in the CourseWeb. Submit your `Permutation.java` to the CourseWeb under this lab by the due date. **No late submission will be accepted**.