CS200: Programming I **Instructor:** Sam Vinitsky

P9: Pierre

(20 points)

Due: Thursday, July 25 @ 11:59pm

Part 0: Introduction

Brrrring brriiiiiing the phone rings – it's Pierre, local layabout. Pierre spends his days lounging around, reading, and getting into unnecessary arguments.

Recently, Pierre was reading through an old loose-leaf copy of *War and Peace*, but unfortunately he dropped it while at a gala, and the pages went flying everywhere! Pierre has recovered some of the pages, but they are in a totally mixed up order! Since the book is so long, it would be hard for him to sort the pages back into the correct by hand, so he has asked for **your** help!

Write a program called `Pierre.java`, and include in it the functions described in this document. Each function will take as input an array of integers called `pages` that represents the current ordering of the pages (where each element is a page number). For example, if the `pages` array were `{10, 4, 5}`, that would mean that the first page in the current ordering is page 10, the second is page 4, and the third is page 5. In the `pages` array, all page numbers are positive (non-zero), some pages numbers may be missing, and no page number can appear twice.

The `pages` array is *in order* if each page number is greater than the page number that came before it. For example, the array `{5, 6, 7, 8, 9, 11}` is in order, despite missing several pages. The array `{1, 2, 3, 4, 10, 8}` is *not* in order – the last two pages should be swapped.

You do not need to write any special code in the main of `Pierre.java`, but you should use it to test your functions for a variety of inputs. We will not be grading this directly, but if there are issues with your program, we may consult this to see how thoroughly you tested your functions.

Your program may not use any built-in Java classes. This includes Random, Math, Scanner, and any static methods from the Arrays or String classes. If you have questions about what you are/not allowed to use, please ask on Piazza.

Each file you turn in should include a comment at the top with (1) your full name, (2) your student ID number, (3) your netID, and (4) the name of anyone you discussed the homework with (excluding Sam and Alex).

As always: **start early, ask questions, and have fun!**

Part 1: Are the pages in order? [4 points]

Before he attempts to reassemble the pages, Pierre would like to have a way to quickly check whether the pages are in order or not. This will help him determine when to **stop** sorting the pages.

Write a function called `isInOrder` that takes as input an array of integers called `pages` representing the current ordering, and returns `true` or `false` depending on whether the pages that are present are in the correct order. This function **must not alter the original pages array,** and must not print anything.

Make sure to test that your function works on a variety of inputs – arrays that are already in order, those that are in completely backwards order, those that are of odd length, those that are only slightly out of order, etc...

Hint: *This function can be written with just a single loop.*

Part 2: Fix the page order! [8 + 2 + 3 points]

Pierre would like you to reassemble the pages in the correct order. Write a function called `order` that takes as input an integer array called `pages` representing the current ordering, and changes this array to be in order. **Notice that you must change the original array.** Your function should not return or print anything.

Before writing this function, I **highly** recommend discussing your approach with a friend, or with Alex or Sam. You should write pseudocode, and walk through several examples to make sure your algorithm makes sense.

Make sure to test that your function works on a variety of inputs – those that are already in order, those that are in completely backwards order, those that are of odd length, those that are only slightly out of order, etc... *The correctness of your function is worth 8 points.*

We will be grading the efficiency of your function. It must properly order any array of length less than 100,000 in **less than a minute**. *The efficiency of your function is worth 2 points.*

In a separate file called `Explanation.txt`, write pseudocode[1] for your algorithm. Then, explain

[1]This can be very vague pseudocode. You can say things like "if the list is sorted", "swap the smallest element into the first spot", "first the largest element in the list", etc

with words how the algorithm you designed works, and **why it is correct**. There is no length requirement for this explanation, but if your explanation itself is less than 3 sentences, you are likely doing something wrong. *This explanation is worth 3 points.*

Hint: *In order to develop an algorithm to solve this problem, I would recommend working through how* ***you*** *would order a list of numbers, and then develop this into an automated algorithm. I would recommend doing this as follows: write the numbers 1-10 on small slips of paper, and place them in a row in a random order. Then, manually sort them into the correct order. Think about* ***how*** *you did this, and then write pseudocode that follows this same process. Finally, translate this pseudocode into Java code.*

Part ∞: Feedback Form [3 points]

Fill out this feedback form **after you submit** this assignment. Completion of this will count towards your grade, but your responses themselves will not affect your grade in any way (so be honest!).

What to turn in

On Canvas, turn in a zip folder named `<your_net_id>_P9.zip` that contains the following file:

- `Pierre.java` [17 points total]
  - `boolean isInOrder(int[] pages)` (4 points)
  - `void order(int[] pages)` ($8 + 2 + 3$ points)
- `Explaination.txt` [3 points]

Also complete the feedback form. [3 points]
