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

P8: Leo at the Parade

(25 points)

Due: Monday, July 22 @ 11:59pm

Part 0: Introduction

Brrrring brriiiiiing the phone rings – it's Leo, local factory owner! Leo is tired of the drudgery of everyday factory life, and has decided to take a day off to attend the Memorial Day parade. This parade consists of a series of **floats of various heights**. Leo wants to make the most of his day off, so he wants to make sure he is seeing the most impressive portions of the parade. That's where you come in!

In order to help Leo out, you should write a program named `Parade.java` that includes the functions described in this document. **The input to each function will be an integer array `heights` representing the heights of each float in the parade, in the order they appear.** Each float's height is a positive (non-zero) integer, and repeated heights are allowed. For example, if the `heights` array was `{10, 5, 11, 1}`, this would mean that the first float will be 10 feet tall, the second float will be 5 feet tall, the third float will 11 feet tall, and the last float will be 1 foot tall. You can always assume there are at least 3 floats in the parade.

For each function that you write, make sure the signature **exactly** matches the specs, or else you will receive **no points** for that function. In the main of your `Parade.java`, you should write tests to make sure that your functions are working (these will not be graded in any way).

Before attempting this problem set, make sure you are comfortable with the practice problems from this week's lectures, available here.

As usual, you may not use anything we have not covered in class. This includes built-in static Array class methods, String methods, anything involving ArrayLists/Math/Random, and anything else we haven't explicitly covered in class. If you have questions about what you are/not allowed to use, ask on Piazza.

Each program 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: Finding the Best Float [4 points]

Leo is very short, so he is easily impressed by **tall** floats. But two tall floats in a row are boring, since the second float doesn't seem quite as tall compared to the one that came before it... What makes Leo happiest is seeing a short float followed by a much taller float. This difference in height excites Leo, since the second float seems **so** much taller than the first one!

Leo refers to the difference in height between float `i` and `i-1` as the *apparent height* of float `i`. Note that this could be negative, if float `i` is shorter than float `i-1`... The apparent height of the first float is always 0, since there is no float before it to compare it to! For example, consider the array `{4, 10, 1, 8}`. The apparent heights of the floats are `0`, `6`, `-9`, and `7`, in order.

Leo wants to know which float is in the parade has the **largest apparent height**. Write a function called `findBestFloat` that takes an input an array of integers representing the heights of the floats and returns the **index** of the float with the largest apparent height. If multiple floats have the same apparent height, pick the one that appears **first**.

The signature of your function must be exactly:

```
public static int findBestFloat(int[] heights)
```

You function should **return** a value, and should not **print** anything. Make sure your function works by testing it with a variety of different inputs. A **small sample** of input/output pairs are listed here for reference, but you should test more than just these...

An input of `{4, 5, 2, 5, 4}` should return `3`
An input of `{1, 5, 10, 1}` should return `2`
An input of `{1, 10, 22, 40, 100, 40000, 40001}` should return `5`
An input of `{1000, 1001, 1002, 1005}` should return `3`
An input of `{5, 4, 3, 2, 1}` should return `0`
An input of `{1, 1, 1, 1}` should return `0`

Part 2: Finding the Best Average Height [5 points]

Leo is a complex man. While he is excited by large changes in height, he is also excited by consistency. He would like to know **which 5 floats in a row have the largest average height**. Write a function called `findTallestGroup` that takes an input an integer array representing the heights of the floats, and returns the **index of the middle float** in the group of 5 with the largest average height. If multiple groups of 5 have the same average, pick the one that appears **first**.

The signature of your function must be exactly:

```
public static int findTallestGroup(int[] heights)
```

You function should **return** a value, and should not **print** anything. Make sure your function works by testing it with a variety of different inputs. A **small sample** of input/output pairs are listed here for reference, but you should test more than just these...

An input of `{5,2,4,4,100,100,100,100,100,2,3,4,2,3}`, should return `6`
An input of `{500,100,200,300,200,4,3,2,1,500,200,300,100,200}`, should return `2`
An input of `{500,100,200,300,200,4,3,2,1,500,200,300,100,201}`, should return `11`

For this problem, you can assume that all parades have at least 6 floats in them.

Part 3: Finding the Novel Floats [6 points]

Leo likes excitement. To Leo, the most exciting thing is seeing a float that is taller than **all** of the floats that came before it. This makes Leo jump for joy!

Write a function called `findNovelFloats` that takes as input an integer array, and **prints out** the index of each float that is taller than all the floats that came before it. Each index should be printed on its own line, and the indices should be printed in increasing order. The very first float **does** count for this, since it will always be taller than all those that came before it... The signature of your function must be exactly:

```
public static void findNovelFloats(int[] heights)
```

You function should **print out** several lines, and not **return** anything. Make sure your function works by testing it with a variety of different inputs. A **small sample** of input/output pairs are listed here for reference, but you should test more than just these...

An input of `{10, 5, 6, 7, 8, 11}` should print out `0` and `5`, each on their own line
An input of `{1, 2, 3, 4, 5, 5}` should print out `0`, `1`, `2`, `3`, and `4`, each on their own line
An input of `{10, 9, 10}` should print out `0`
An input of `{2, 5, 4, 8, 6, 7, 8, 10}` should print out `0`, `1`, `3`, `7`, each on their own line

Part 4: Finding the Longest Increasing Sequence [7 points]

More than anything, Leo enjoys a an extended sequence of increasing tension. As mentioned before, Leo enjoys when a float is taller than the float before it. It is even more exciting when this happens multiple times in a row! That is, if float 5 is taller than float 4, which was taller than float 3. Leo wants to know the **longest sequence of floats** where each is taller than the last. Leo doesn't care how **much** the floats' heights increase, merely that they do.

Write a function called `findLongestIncreasingSequence` that takes as input an integer array representing the heights of the floats, and returns an **array of length 2**. The first element of this array should be the starting index of the longest increasing sequence of heights, and the second element should be the last index in the longest increasing sequence of heights. If there are multiple increasing subsequences of the same length, pick the first one.

The signature of your function must be exactly:

```
public static int[] findLongestIncreasingSequence(int[] heights)
```

You function should **return** a value, and should not **print** anything. Make sure your function works by testing it with a variety of different inputs. A **small sample** of input/output pairs are listed here for reference, but you should test more than just these...

An input of `{10, 5, 6, 7, 8, 11}` should return `{1,5}`
An input of `{1, 2, 5, 4, 6, 4, 7, 8, 9, 10, 11, 12}` should return `{5,11}`
An input of `{1, 2, 3, 4, 3, 4, 5, 4, 5, 6}` should return `{0,3}`
An input of `{1, 100, 1000000, 2, 3, 4, 5, 1, 2}` should return `{3,6}`
An input of `{10, 9, 8, 7, 6, 5, 4, 3, 2, 1}` should return `{0,0}`

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>_P8.zip` that contains the following file:

- `Parade.java` [22 points total]
  - `public static int findBestFloat(int[] heights)` (4 points)
  - `public static int findTallestGroup(int[] heights)` (5 points)
  - `public static void findNovelFloats(int[] heights)` (6 points)
  - `public static int[] findLongestIncreasingSequence(int[] heights)` (7 points)

Also complete the feedback form. [3 points]