Assignment 1

Description

The purpose of this assignment is to design, implement, and test five sorted array manipulation methods.

Implementation Requirements

This section describes the requirements for the methods you need to implement in a Java class called `SortTools`. **Preconditions** will always be satisfied before we call your methods, and as such you do not need to check or account for them. You must satisfy the worst-case runtime **complexity** requirement.

You must not use the static methods found in `java.util.Arrays` or anything from Java Collections in your solution, except for `Arrays.copyOf` and `Arrays.copyOfRange`. You must not use code from the internet.

- `public static boolean isSorted(int[] x, int n)`
  - Complexity: ***O(n)***
  - Preconditions:
    - `0 <= n <= x.length`
    - `x != null`
  - Behavior:
    - Returns `true` if the first `n` elements of `x` are sorted in non-decreasing order, returns `false` otherwise.
    - If `x.length == 0` or `n == 0`, return `false`.
  - Postconditions:
    - The contents of `x` must not be modified.
  - Example input/output:
    - Arguments: `x` = `[1, 3, 3, 7, 2]`, `y` = `4`
    - Return value: `true`
    - Post-execution state of `x`: `x` = `[1, 3, 3, 7, 2]`
- `public static int find(int[] x, int n, int v)`
  - Complexity: ***O(log(n))***
  - Preconditions:

        - `isSorted(x, n) == true`
        - `0 < n <= x.length`
        - `x != null`
    - Behavior:
        - If `v` is contained within the first `n` elements of `x` return an index of `v` defined as any index **k** such that `k < n` and `x[k] == v`. Otherwise, return `-1`.
    - Postconditions:
        - The contents of `x` must not be modified.
    - Example input/output:
        - Arguments: `x` = `[1, 3, 3, 7]`, `n` = `4`, `v` = `3`
        - Return value: `1` or `2` (either would be correct)
        - Post-execution state of `x`: `x` = `[1, 3, 3, 7]`
- `public static int[] copyAndInsert(int[] x, int n, int v)`
    - Complexity: ***O(n)***
    - Preconditions:
        - `isSorted(x, n) == true`
        - `0 < x.length`
        - `0 <= n <= x.length`
        - `x != null`
    - Behavior:
        - Return a new array `y` with the following properties:
            - `isSorted(y, y.length) == true`
            - If the first `n` elements of `x` contain `v`, `y` contains only the first `n` elements of `x`.
            - Otherwise, `y` contains the first `n` elements of `x` and a new value `v`.
    - Postconditions:
        - The contents of `x` must not be modified.
    - Example input/output 1:
        - Arguments: `x` = `[1, 3, 5, 7]`, `n` = `3`, `v` = `4`
        - Return value: `[1, 3, 4, 5]`
        - Post-execution state of `x`: `x` = `[1, 3, 5, 7]`
    - Example input/output 2:
        - Arguments: `x` = `[1, 3, 3, 7]`, `n` = `3`, `v` = `4`
        - Return value: `[1, 3, 3, 4]`
        - Post-execution state of `x`: `x` = `[1, 3, 3, 7]`
- `public static int insertInPlace(int[] x, int n, int v)`
    - Complexity: ***O(n)***
    - Preconditions:
        - `isSorted(x, n) is true`
        - `0 < n < x.length`
        - `x != null`
    - Behavior:
        - If `x` does not contain `v`, modify `x` such that:
            - `isSorted(x, n + 1) == true`
            - the first `n + 1` elements of `x` contain `v`

            - `x` contains exactly one `v`
        - If `x` contained `v` within the first `n` elements, then return `n` otherwise return `n + 1`.
    - Postconditions:
        - `x` is modified as described above.
        - The contents of `x` beyond the first `n + 1` elements (or `n` elements, if `v` was not inserted) can be arbitrary values. You may preserve them or modify them.
    - Example input/output 1:
        - Arguments: `x` = `[1, 3, 5, 7, ...]`, `n` = `3`, `v` = `4` (`...` can be any quantity of integer values)
        - Return value: `4`
        - Post-execution state of `x`: `x` = `[1, 3, 4, 5, ...]`
    - Example input/output 2:
        - Arguments: `x` = `[1, 3, 3, 7, ...]`, `n` = `3`, `v` = `4`
        - Return value: `4`
        - Post-execution state of `x`: `x` = `[1, 3, 3, 4, ...]`
- `public static void insertSort(int[] x, int n)`
    - Complexity: ***O(n^2)***
        - In the general case, your function must have ***O(n^2)*** time complexity. When `x` is nearly sorted, your function must have ***O(n)*** time complexity.
            - The formal definition of nearly sorted is:

              `x[k] <= x[k+1]` for all `0 <= k < n`

              except for at most **C** values of **k** (where **C** is a constant).
        - Informally, your function must have linear time complexity if all the elements in `x` are sorted with just one value out of place. You have choices of algorithm based on the above criteria alone, but we want you to implement **insertion sort**.
    - Preconditions:
        - `0 < x.length`
        - `0 < n <= x.length`
        - `x != null`
    - Behavior:
        - Sort the first `n` elements of `x` in non-decreasing order.
    - Postconditions:
        - `x` has its first `n` elements sorted in non-decreasing order.
        - The other elements of `x` are unmodified.
    - Example input/output 1:
        - Arguments: `x` = `[7, 3, 1, 3]`, `n` = `3`
        - Return value: N/A
        - Post-execution state of `x`: `x` = `[1, 3, 7, 3]`
    - Example input/output 2:
        - Arguments: `x` = `[7, 3, 1, 3]`, `n` = `1`
        - Return value: N/A
        - Post-execution state of `x`: `x` = `[7, 3, 1, 3]`

Additional Requirements

Testing

You have been provided 2 JUnit test cases and a testing script. You must ensure that your code passes these test cases. It must also pass the testing script on the ECE Linux 64-bit machine kamek. We will release instructions for running the script on Piazza.

The above tests are not comprehensive, so we recommend that you also write your own JUnit tests to verify that your methods work correctly. Alternatively, you can create and use a `main()` method.

You may post ideas about testing to Piazza. Do not post test case code or solution code. Your test case ideas should not hint at the solution algorithm.

General Requirements

No exceptions will be made for not following these requirements.

- You must use good style, including indentation, variable and method names, spacing, and comments.
- You must ensure that your program compiles and runs with Java 8.
- You must complete the header by copying and filling out the template found in Header.txt onto every file you submit. No need to submit Header.txt, you can safely delete it afterwards.
- You must not delete or modify the package statement found at the top of every starter code file.
- You must make sure that the test cases you are given pass with your code, using JUnit testing. Note: passing these tests does not guarantee a perfect score.
- You must test your program further. You can do this via JUnit or the main method. Do not submit Main.java, SampleTest.java, or any other test files you write. Note that SortTools.java may NOT have a main method.
- You must run your code on the ECE Linux 64-bit machine kamek.
- You must run the provided sample grading script on your code.
- You must download your program from Canvas after submitting and test it again with the sample grading script to confirm the submitted file is in the correct format.
    - Getting your code to pass the script is not optional, and code submitted without completing this step may earn no points. Try out the script a day before the due date to identify any issues early.

Deliverables

All deliverables must be submitted to the corresponding Canvas assignment **before the deadline**.

Canvas assignment: 1. Sort Tools

- Project1_EID.zip
    - Zip file structure:

```
Project1_EID.zip (zipped file)
    assignment1/ (folder)
        SortTools.java
```

  - Do not include your test files.
  - Do not include any .class files.

How to Create the Submission File on Linux

When you're developing your project, your SortTools.java file will be in a directory called assignment1/, which is the package directory. assignment1/ will probably be in a directory called src/ in your Eclipse workspace. Copy assignment1/ over to a temporary location, remove everything but SortTools.java from it, and then zip up the assignment1 directory/folder. On Linux, you can do this by changing to the directory containing assignment1/, then running:

`zip -r Project1_EID.zip assignment1/`

This will create a new file called Project1_EID.zip. Unzip it (`unzip Project1_EID.zip`) to make sure that there is an assignment1/ directory in there, and that SortTools.java is correctly in assignment1/. Run the grading script on it, then you are ready to submit the zip file.

FAQ

1. I re-submitted my project after finding an error and noticed that Canvas automatically changed the file name to Project1_EID-1.zip. Will this be a problem during grading?

- No.

2. I am trying to run the script on OS X/Linux but it says `python2 command not found`. `python --version` shows me Python 2.7.10 installed.

- The grading script is only designed to run on the LRC computers. You likely have a different version of Python installed. Run the script on kamek instead to ensure there are no issues with the script's dependencies. If you want to test on your own computer, use the JUnit test classes directly.

3. I have a weird leftover file in a folder on kamek that cannot be deleted.

- A workaround solution is to run the script in a fresh directory. Come to office hours and a TA may be able to help you remove it.

4. What score should I get when I pass both the sample tests given?

- 2/2 points.

5. Am I allowed to share test cases with other students?

- No, you are not allowed to share test cases for this particular assignment. For later ones, sharing is allowed.

6. Do we need to account for preconditions in our code? Or can we assume that they will always be met when you test our code?

- There is no need to check whether the preconditions are true.