# EE 422C Project 1
## Arrays and Sorting
## Individual Assignment

*See Canvas for due dates and further instructions.*

### General Assignment Requirements

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

The following 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, as such you do not need to check or account for them. The top right corner of each description lists the worst case runtime complexity of the method. You must satisfy this requirement.

### Methods

**boolean isSorted(int[] x, int n)** $O(n)$

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**.

Preconditions
- **0** <= **n** <= **x.length**
- **x** != **null**

The contents of ***x*** must not be modified.

**int find(int[] x, int n, int v)** $O(log(n))$

If ***v*** is contained within the first ***n*** elements of ***x***, return an index of ***v***. If ***v*** is not contained within the first ***n*** elements of ***x*** return **-1**.

If more than one ***v*** is present in ***x*** then the function must return the index of one but may return any index ***k*** such that ***k*** < ***n*** where ***x***[***k***] == ***v***.

Preconditions
- isSorted(**x**, **n**) is **true**
- **0** < **n** <= **x.length**
- **x** != **null**

The contents of ***x*** must not be modified.

**int[] insertGeneral(int[] x, int n, int v)** $O(n)$

Return a new array ***y*** with the following properties
- isSorted(**y**, **y.length**) is **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***

Preconditions
- isSorted(**x**, **n**) is **true**
- **0** < **x.length**
- **0** <= **n** <= **x.length**
- **x** != **null**

The contents of ***x*** must not be modified.

**int insertInPlace(int[] x, int n, int v)** $O(n)$

If ***x*** does not contain ***v*** modify ***x*** such that
- isSorted(**x**, **n + 1**) is **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**.

Preconditions
- isSorted(**x**, **n**) is **true**
- **0** < **n** < **x.length**
- **x** != **null**

The contents of ***x*** beyond the first ***n*** or ***n + 1*** elements are don't care. You may preserve them or modify them.

**void insertSort(int[] x, int n)** $O(n^2)$*

Sort the first ***n*** elements of ***x*** in non-decreasing order.

Preconditions
- **0** < **x.length**
- **0** < **n** <= **x.length**
- **x** != **null**

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

$$x[k] \leq x[k+1] \text{ for all } 0 \leq k < n$$

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**.

## Testing

You have been provided a file containing 2 JUnit test cases and a testing script. You must ensure that your code runs with these test cases and script on the ECE Linux 64-bit machines, such as kamek. Instructions for running the script can found on Piazza.

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.

## Additional 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 machines, such as Kamek.
- You must run your code on the provided sample grading script.
- You must submit your program to Canvas before the deadline. Upload SortTools.java to the corresponding assignment on Canvas.
- You must download your program from Canvas after submitting. Make a new project in Eclipse or your preferred IDE and compile, run, and test the file you downloaded.

**FAQ**

1. I re-submitted my SortTools.java file after finding an error and noticed that Canvas automatically changed the file name to SortTools-1.java.
   Will this be a problem during grading?
   a. 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.
   a. We intend for the script to be run from the ECE LRC machines for reasons such as this. Many Linux distributions will have an explicit python2 binary due to there being Python 2 and Python 3 around. If you really want to run it locally, go into the perform_grading.sh file and change the instance of "python2" to just "python". Make sure you have the other dependencies like Maven installed.
3. I have a weird leftover file in a folder on kamek that cannot be deleted.
   a. A workaround solution is to run the script in a fresh directory. Come to office hours a TA may be able to help you remove it.
4. May I have System.exit() in my code?
   a. No. It breaks the JUnit testing script.
5. What score should I get when I pass both the sample tests given?
   a. helper_scripts/score_extractor.py assigns a weight of 2 points to one JUnit test and 3 points to the other test, so passing them both means getting a grade of 5.
6. Am I allowed to share test cases with other students?
   a. No, you are not allowed to share test cases for this particular assignment. For later ones, sharing is allowed.