You are required to complete weekly exercises as part of the 20% allocated to ongoing CA. These exercises will be linked to the week's topic and by completing them, along with your unmarked exercise sheets, you will gain proficiency in programming.

NOTE 1: EVERY METHOD MUST INCLUDE A JAVADOC COMMENT AS DOCUMENTATION.

Note 2: You MAY NOT USE any built-in methods to carry out the logic of these algorithms other than when specifically stated.

Sheet 1: Brute Force Actions

The following exercises must be given the exact names listed, and must be placed in the exact packages stated. If this is not done, the tests will fail.

Set up:

Before you attempt the exercises, do the following:

1) Create a new project – this will be used solely for your ongoing CA work. The project name should be: the year, the module name and your name, e.g. 2025_Algorithms_MichelleGraham.
2) Preparation for git tracking/source control:
    a. Add a git repository to your project
    b. Share your project to github.
    c. Add me (**mgraham-dkit**) as a collaborator on github. There will be a short screencast showing how to do this in the CA section of moodle.
3) Preparation for testing:
    a. Copy the text from the dependencies.txt file (found in the submission point) and paste it into your pom.xml file. This text should be placed after the closing properties tag (</properties> and before the closing project tag (</project>)
    b. Create a package called **`utils`** in your project's main -> java folder.
    c. Create a package called **`utils`** in your project's test -> java folder.
    d. Add the test files supplied in the submission point to the **`utils`** package in the test folder.

Before you begin:

- Create a class called **`NumberUtils.java`** in the **`utils`** package.
- Create an application called **`Statistics.java`** in a package called **`ui`**.

Reminder 1: When writing JavaDocs, ensure that you:

- Explain what the method does (one line description)
- Explain what the parameters are used for (@param tags) <- if the method takes in a parameter. There should be an @param tag for every parameter you include
- Explain what the method returns (@return tag) <- if the method returns a value

Reminder 2: When writing your code, commit and push when:

- The method is functional (your commit for this should use the "feat" label) and passing tests
- The JavaDoc has been added (your commit message for this should use the "docs" label)
- A feature in your application has been added

Exercise 1:

Write a static method called `getPos()` in `NumberUtils` that finds the first position of a supplied int in an array.

- Parameters: An array of ints and a single int
- Returns: an int (the position)

Write a JavaDoc for this method.

Remember to commit and push your code after your method is completed and after your Javadoc is added.

Exercise 2:

Write a static method called **`getLastPos()`** in **`NumberUtils`** that finds the **last** position of a supplied int in an array.

- Parameters: An array of ints and a single int
- Returns: an int (the position)

Write a JavaDoc for this method.

Remember to commit and push your code after your method is completed and after your Javadoc is added.

Exercise 3:

Write a static method called **`isIdentical()`** in **`NumberUtils`** that determines if two supplied arrays of ints are identical (the same data in the same order).

- Parameters: Two arrays of ints
- Returns: a boolean

Write a JavaDoc for this method.

Remember to commit and push your code after your method is completed and after your Javadoc is added.

Exercise 4:

Write a static method called **`isEqual()`** in **`NumberUtils`** that determines if two supplied arrays of ints contain the same information (this can be in the same or different order). **Note: You MAY use Arrays.sort() to carry out this logic.**

- Parameters: Two arrays of ints
- Returns: a boolean

Write a JavaDoc for this method.

Remember to commit and push your code after your method is completed and after your Javadoc is added.

Exercise 5:

Write a static method called **`isSubset()`** in **`NumberUtils`** that determines if the first supplied arrays of ints is a subset of the second supplied array (all data from one array is found somewhere in the other). Note: A subset cannot contain the same number of elements as the original set.

- Parameters: Two arrays of ints, (A, B) (your method should check if A is a subset of B)
- Returns: a boolean

Write a JavaDoc for this method.

Remember to commit and push your code after your method is completed and after your Javadoc is added.

Exercise 6:

Write a static method called **`getMax()`** in **`NumberUtils`** that returns the largest number in a supplied array.

- Parameters: An array of ints and a single int
- Returns: an int (the position)

Write a JavaDoc for this method.

Remember to commit and push your code after your method is completed and after your Javadoc is added.

Exercise 7:

Add the following logic to `Statistics.java` class (this should be added in a `main()` method):

- Create a hard-coded array containing the following: {10, 20, 10, 20, 60, 10, 5, 60, 5, 60}
- Using the code supplied on moodle (**`getRandomArray()`** in DataGenerator), generate an array of random ints.
- Once this is done, your program should:
  - Display the contents of both arrays.
  - Find the largest value in the random array.
  - Find the position in which this value first appears in the array.
  - Find the position in which this value last appears in the array.
  - Determine whether or not the hard-coded and generated arrays are identical, equal or if one is a subset of the other.

Remember to commit and push your code after each feature has been added to the program.

Deliverables:

A github link to your project should be uploaded to moodle. Make sure you have added me to the project as a collaborator!

Grading:

All code must be your own work, and you may be examined "in class" at any point during semester. All input and output should adhere to exercise specifications. **The use of AI-generated code (including (but is not limited to) Claude, Github/Microsoft Co-pilot, in-line completion in IntelliJ and OpenAI's ChatGPT) is not permitted.**

Testing:

You have been supplied with junit tests to confirm your code performs correctly under a number of circumstances, i.e. not just with correct input. Make sure to run these tests as you develop your code, so you can spot any possibilities you might have missed and adjust accordingly. The tests have been split into separate files for separate methods, so you can work on one at a time without needing to comment tests out/remove them from the runner.