Homework 4

This homework practices working with arrays, and with strings. Unlike previous homework assignments where each question was in a separate file, you will now organize your code into two classes: `ArrayOps` and `StringOps`. Each class will contain a set of typical functions related to array and string operations, respectively.

We provide two skeletal class files, `ArrayOps.java` and `StringOps.java`. Each class has a `main` function. For each function that you develop, we recommend writing test code that tests that function. Put your test code in the supplied `main` functions. We will not run the `main` functions that you will write, and will not grade them – it's for your own testing only. In other words: Start by testing your work locally, on your PC, using the test code that you write. When you feel that your classes are well written, you can start the commit and submit process, which we describe at the end of this document.

Note: Your GitHub repository will also include two other classes `TesterArrayOps.java` and `TesterStringOps.java` that we wrote, which also contain `main` functions. Technically speaking, these two classes have nothing to do with the main functions that you will write. You should not touch these files, since editing them will mess up the operations of the autograders.

In addition to the functions that you *have* to implement as part of this homework, you *can* also write some additional helper functions and add them to the relevant class, in order to simplify your code. Your helper functions (if you write any) will not be tested or graded by the autograders.

ArrayOps

This class offers some functions for processing arrays:

- findMissingInt
- secondMaxValue
- containsTheSameElements
- isSorted

1. findMissingInt

This function takes a single parameter: an array of integer values. Let us assume that the length of this array is n. The array contains each integer $0 \leq x \leq n$, exactly once, except for one of these integers, which is missing. The integers are stored in the array in no particular order. The function returns the missing integer. Assume that the input is valid (obeys the rules described above), and that the array's length is ≥ 1.

Examples:

```
findMissingInt(new int[] {3, 0, 1});            // 2
findMissingInt(new int[] {0, 1, 2, 3, 4, 6});  // 5
```

```
findMissingInt(new int[] {0});                  // 1
```

2. secondMaxValue

This function takes an array of integers, and returns the second largest number in the array. The numbers in the array may appear more than once. Assume that the array's length is > 1.

Examples:

```
secondMaxValue(new int[] {6, 9, 4, 7, 3, 4});    // 7
secondMaxValue(new int[] {1, 2, 3, 4, 5});       // 4
secondMaxValue(new int[] {2, 8, 3, 7, 8});       // 8
secondMaxValue(new int[] {1, -2, 3, -4, 5});     // 3
secondMaxValue(new int[] {-202, 48, 13, 7, 8});  // 13
```

3. containsTheSameElements

This function takes two arrays of integers, and returns true if both arrays contain the same numbers. The number of occurrences of each number, and their order, are not important (note: this is similar to checking the identity of *sets* (קבוצות) in mathematics).

Examples:

```
containsSameElements(new int[] {1, 2, 1, 1, 2}, new int[] {2, 1});               // true
containsSameElements(new int[] {2, 2, 3, 7, 8, 3, 2}, new int[] {8, 2, 7, 7, 3}); // true
containsSameElements(new int[] {1, 2, 3}, new int[] {1, 2, 3});                  // true
containsSameElements(new int[] {3, -4, 1, 2, 5}, new int[] {1, 3, -4, 5});       // false
```

4. isSorted (Is sorted)

This function takes an array of integers and checks if the array is sorted increasingly, from the minimum value to the maximum value, or decreasingly, from the maximum value to the minimum value.

Examples:

```
isSorted(new int[] {7, 5, 4, 3, -12});  // true
isSorted(new int[] {1, 2, 3});          // true
isSorted(new int[] {1, -2, 3});         // false
isSorted(new int[] {1, 1, 500});        // true
isSorted(new int[] {1, 3, 2});          // false
```

StringOps

The StringOps class will offer some string processing functions:

- capVowelsLowRest
- camelCase
- allIndexOf

Your implementation (the Java code that you write and submit) can use only use the following String methods:

- `str.charAt(int i)`
- `str.length()`
- `str.indexOf(char ch)`
- `str.substring(int start)`
- `str.substring(int start, int end)`

1. capVowelsLowRest

This function takes as input a string containing only letters, organized into words that are separated with a single space (there is no need to write code that checks that the input is valid). The function returns a string in which all the English vowels (*a,e,i,o,u*) in the original strings are changed to uppercase, and all the other characters are changed to lowercase (if they were lowercase in the original string, they remain the same).

Examples:

```
capVowlesLowRest("Hello World")          // "hEllO wOrld";
capVowlesLowRest("One two tHRee world")  // "OnE twO thrEE wOrld";
capVowlesLowRest("vowels are fun")       // "vOwEls ArE fUn";
capVowlesLowRest("intro")                // "IntrO";
capVowlesLowRest("yellow")               // "yEllOw";
```

2. camelCase

This function also takes as an input a string containing only letters, organized into words that are separated with a single space (there is no need to write code that checks that the input is valid). The function returns a string in which the original string is modified according to three rules:

1. The first word is changed to lowercase (if it's not lowercase already).
2. The first letter of each word is changed to uppercase (if it's not uppercase already), and all the remaining letters in the word are changed to lowercase (if they are not lowercase already).
3. All the spaces (if any) are removed.

Examples:

```
camelcase("Hello World")                          // "helloWorld"
camelcase("HELLO    world")                       // "helloWorld"
camelcase(" tWo     wordS")                       // "twoWords"
camelcase("world")                                // "world
camelcase("   Intro to  coMPUter   sCIEncE ")     // "introToComputerScience"
```

3. allIndexOf

This function receives two parameters: A string and a character. The function returns an array containing all the indexes in which the character appears in the string.

Assume that the given string is not empty, and that the given character appears in the string at least once.

Examples:

```
allIndexOf("Hello world",'l');          // output: {2, 3, 9}
allIndexOf("Hello worLd",'l');          // output: {2, 3}
allIndexOf("Hello world",'o');          // output: {4, 7}
allIndexOf("Hello world",' ');          // output: {5}
allIndexOf("Hello world",'d');          // output: {10}
allIndexOf("MMMM",'M');                 // output: {0, 1, 2, 3}
```

Hint: You have to go through the given string twice. Once for determining the size of the array that you have to declare and return, and once for populating this array with values.

Submission Instructions for Git Classroom

Accepting the Assignment

Start by accepting the assignment through the link to Github provided on Moodle. This action will automatically create a repository in your GitHub Classroom account.

Cloning the Repository

Clone this repository to your local machine. For more information, watch the video we prepared on how to clone a repository.

Committing and Pushing Changes

1. Work on your assignment on your local machine by editing the files `StringOps.java` and `ArrayOps.java`. Commit the changes to your local repository.
2. Once ready to submit your work, push these changes to the GitHub Classroom repository.
3. You can view your submission, feedback, and grades directly within GitHub Classroom.
4. You may repeat this process as many times as you wish in order to improve your code before the deadline.
5. **Avoid modifying other files within the repository!** This will interfere with the functioning of the auto-grading system. You should edit and work only on the two files `StringOps.java` and `ArrayOps.java`.

Submission deadline: February 16th, 2024, 23:55.