Project 1

Due Date: Friday 9 September 2022 by 11:59 PM

General Guidelines.

The instructions given below describe the required methods in each class. You may need to write additional methods or classes that will not be directly tested but may be necessary to complete the assignment.

In general, you are not allowed to import or use any additional classes in your code without explicit permission from your instructor!

Note: It is okay to use the Math class if you want.

Note on academic dishonesty: Please note that it is considered academic dishonesty to read anyone else's solution code, whether it is another student's code, code from a textbook, or something you found online. You MUST do your own work! It is also considered academic dishonesty to share your code with another student. Anyone who is found to have violated this policy will be subject to consequences according to the syllabus and university policy.

Project Overview.

There are two parts to this project. Both parts make use of the *Array.java* class, which is a wrapper class for an integer array. You are required to use this class as it is. The purpose is because one of the major objectives of this project is to get you thinking not only about whether or not a solution works but also how efficiently it works. To that end, the Array class counts the number of times you access the array, which will help the graders in measuring the efficiency of your approach.a

Note: It is considered academic dishonesty to try to trick the auto-grader or the human graders into thinking your solution is more efficient than it is. If you're ever unsure of what that means, ask!

Part 1.

Problem Statement: **Given an array of integers *A* and an integer *m*, determine the maximum product of *m* consecutive integers in the array.**

Example:
A = [3, 1, 6, 2, 8, 9, 3, 4], m = 3
Answer: 216 (which is the product of 8, 9, and 3)

Guidelines:

- Implement your solution in a class called *Part1.java* with the following method signature:
  - `public static int maxProduct(Array a, int m)`
  - The method takes the Array and the integer and returns the maximum product.
- You are not allowed to use any additional data structures.
- For full credit, your solution should be no worse than *O(N+M)* where *N* is the number of elements in *A* and *M* is *m*. You might also want to reduce the constants in front of those as much as possible because a solution that is *O(N+M)* could still include some inefficient things.

Part 2.
Problem Statement: **Given an array of integers *A*, rearrange *A* so that all the 0's are pushed to the end of the array. All other values should stay in the same relative order.**
Examples:
A = [2, 6, 1, 9, 0, 2]
Result: [2, 6, 1, 9, 2, 0]
A = [0, 0, 6, 1, 0, 9, 8]
Result: [6, 1, 9, 8, 0, 0, 0]
A = [3, 1, 4, 1, 5, 9, 2]
Result: [3, 1, 4, 1, 5, 9, 2]

Guidelines:

- Implement your solution in a class called *Part2.java* with the following method signature:
  - `public static void pushZeroes(Array a)`
  - `The method takes the array and pushes all 0's to the end of the array.`
- You are not allowed to use any additional data structures.
- For full credit, your solution should be no worse than *O(N)* where *N* is the number of elements in *A*. You might also want to reduce the constant in front of the *N* as

much as possible because a solution that is *O(N)* could still include some inefficient things.

Testing & Submission Procedure.

I am not providing test code for this project because the actual code for the solutions should be pretty short and it's good practice for you to learn to test your own code. You should test your code for correctness and analyze it for efficiency.

Your code must compile and run on lectura, and it is up to you to check this before submitting.

To test your code on lectura:

- Transfer all the files (including test files) to lectura.
- Run *javac *.java* to compile all the java files.
- Run *java <filename>* to run a specific java file.

After you are confident that your code compiles and runs properly on lectura, you can submit the required files using the following **turnin** command. Please do not submit any other files than the ones that are listed.

```
turnin cs345p1 Part1.java Part2.java
```

Upon successful submission, you will see this message:

```
Turning in:
    Part1.java -- ok
    Part2.java -- ok
All done.
```

Note: Only properly submitted projects are graded! No projects will be accepted by email or in any other way except as described in this handout.

Grading.

Auto-grading

Part	**Total Points**	**Details**
1	30	20 points for correctness 10 points for efficiency
2	30	20 points for correctness 10 points for efficiency

Your score will be determined by the the tests we do on your code minus any deductions that are applied according to the following list of possible deductions.

- Bad Coding Style (up to 10 points)
- Not following directions (up to 100% of the points depending on the situation)
- Late Submission (10 points per day)
- Inefficient code (up to 10 points per part)

If your code does not compile/run on lectura, the TA may give you an automatic 0. If you find that this is due to a small error that is easily fixed, you may fix the code and submit a regrade request to the TA, but projects like this will still get a deduction of at least 15 points.

For regrades, you should contact the TAs directly (email all of them), and you should do so within 72 hours of the grade being released on D2L.