COS 126	Princeton University	Fall 2022

Programming Exam 1

Before the exam. Read this page of instructions before the exam begins. Do *not* start the exam (or read the next page) until instructed to do so.

Duration. Once the exam begins, you have 80 minutes to complete it.

Submission. Submit your solutions on *TigerFile* using the link from the *Exams* page. You may submit multiple times (but only the last version will be graded).

Check Submitted Files. You may click the *Check Submitted Files* button to receive partial feedback on your submission. We will attempt to provide this feature during the exam (but you should not rely upon it).

Grading. Your program will be graded primarily on correctness. Clarity (including comments) and design will also be considered. You will receive partial credit for a program that implements some of the required functionality. You will receive a substantial penalty for a program that does not compile.

Allowed resources. During the exam you may use only the following resources: course textbook, companion booksite, course website, course Ed, your course notes, and your code from the programming assignments. For example, you may not use *StackOverflow* or *Google*.

No collaboration or communication. Collaboration and communication during this exam are prohibited except with course staff. We will be sitting outside the exam room if you have a clarification question.

No electronic devices or software. Software and computational/communication devices are prohibited, except to the extent needed for taking this exam (such as a laptop, browser, and IntelliJ). For example, you must close all unnecessary applications and browser tabs; disable notifications; and *turn off your cell phone*.

Honor Code pledge. Write and sign the Honor Code pledge by typing the text below in the file `acknowledgments.txt`.

> *I pledge my honor that I will not violate the Honor Code during this examination.*

Electronically sign it by typing `/s/` followed by your name.

After the exam. Discussing or communicating the contents of this exam before solutions have been posted is a violation of the Honor Code.

Deliverables. You will submit two Java programs along with an `acknowledgments.txt` file:

1. A library of functions `AppleSharing.java` related to the *apple-sharing game*.
2. A client program `AppleSharingGame.java` that simulates the game on a specific input.

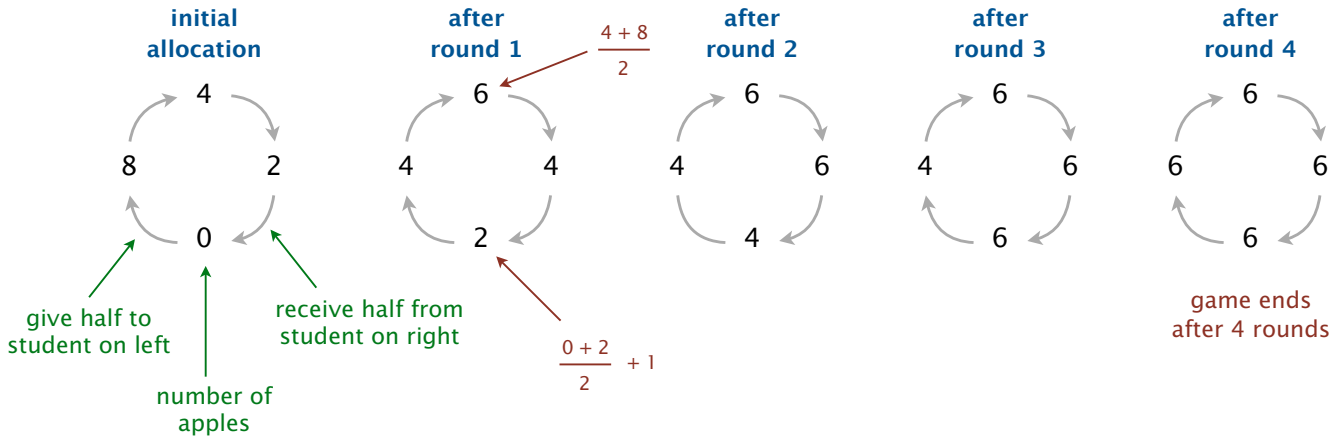
Parts 1 and 2 will be assessed independently, so you may do Part 2 without completing Part 1.

Apple-sharing game. Consider a group of n students, sitting in a circle, and facing their teacher in the center. Each student starts with an even number of apples.

- If all students have the same number of apples, the game ends.
- Otherwise, each student *simultaneously* gives half of their apples to the student sitting on their left (and receives the apples given to them from the student sitting on their right).
- Afterwards, each student that ends up with an odd number of apples receives one extra apple from their teacher.
- This process is repeated until the game ends.

Example. Here is an example with $n = 4$ students, initially with 0, 2, 4, and 8 apples (in counterclockwise order).

- After round 1, the bottom student has 2 apples: 0 (remaining after giving half their apples away) plus 1 (received from student to their right) plus 1 (received from teacher).
- After round 1, the top student has 6 apples: 2 (remaining after giving half their apples away) plus 4 (received from student to their right).
- The game ends after 4 rounds (because all students have the same number of apples).



Part 1 (16 points). Using the template file `AppleSharing.java` provided in the project folder as a starting point, write a library of functions that implements the following API:

`public class AppleSharing`	
`public static boolean isEven(int x)`	*is* `x` *an even integer?*
`public static void show(int[] apples)`	*prints the entries of* `apples[]` *on standard output*
`public static boolean isGameOver(int[] apples)`	*are all entries of* `apples[]` *equal?*
`public static void simulateOneRound(int[] apples)`	*simulates one round of the apple-sharing game, updating* `apples[]`

Here is some more information about the required behavior:

- *Even.* The `isEven()` method should work for all integer arguments. The *even* integers are 0, 2, -2, 4, -4, and so forth.
- *Show.* The `show()` method should print the elements of `apples[]` on one line, with one space between each integer. The line should be terminated with a newline.
- *Array argument.* The elements in the `apples[]` array are stored in counterclockwise order. You may assume that the length of `apples[]` is at least two and that each entry of `apples[]` is even and non-negative.
- *Mutating the argument array.*
  - The `show()` and `isGameOver()` methods must *not* change the elements of `apples[]`.
  - The `simulateOneRound()` method must change the elements of `apples[]`.
- *Main method.* You may (optionally) include a `main()` method (such as the following one) for testing. But we will not grade it.

```
public static void main(String[] args) {
    StdOut.println(isEven(8));      // prints true
    int[] x = { 0, 2, 4, 8 };       // initial allocation of apples
    StdOut.println(isGameOver(x));  // prints false
    show(x);                        // prints 0 2 4 8
    simulateOneRound(x);
    show(x);                        // prints 2 4 6 4
}
```

Part 2 (9 points). Write a client program `AppleSharingGame.java` that simulates the apple-sharing game on a specified initial allocation of apples.

- Read the initial allocation of apples from *standard input* (in the format described below).
- Simulate the apple-sharing game, printing the initial allocation of apples and the allocation of apples after each round (in the output format described below).
- Print the total number of rounds.

You may call the methods defined in Part 1, such as `AppleSharing.show()`, even if you did not implement them correctly.

Input format. The input format contains a positive integer n (the number of students), followed by the initial allocation of apples (in counterclockwise order). The initial allocation of apples consists of n even and non-negative integers, separated by whitespace.

Output format. If the game lasts r rounds, the output format consists of $r + 1$ lines followed by the integer r. Line 0 contains the initial allocation of apples (in counterclockwise order). Line i (for $i = 1, 2, \ldots, r$) contains the allocation of apples after round i. The allocation of apples should be a sequence of integers, separated by spaces.

Example. Here is a sample execution (corresponding to the example on page 2):

```
~/Desktop/apples> more apples4.txt
4   <-------  number of students
0 2 4 8   <-------  initial allocation of apples
                    (in counterclockwise order)


~/Desktop/apples> javac-introcs AppleSharingGame.java

~/Desktop/apples> java-introcs AppleSharingGame < apples4.txt
0 2 4 8   <-------  initial allocation of apples
2 4 6 4   <-------  after round 1
4 6 6 4   <-------  after round 2
6 6 6 4   <-------  after round 3
6 6 6 6   <-------  after round 4
4   <-------  total number of rounds
```