Introduction to Computer Science II
Lab 2
Towers of Hanoi

Time to review method calls and, while we're at it, recursion. Oh, and arrays. And pointers to arrays. Yes, all of that.

1 A classic puzzle

The *Towers of Hanoi* puzzle is a classic example for the use of recursive programming. Use Wikipedia and other online resources to read about the puzzle, if it is unfamiliar; additionally, examine the recursive solutions, available in many programming languages (including Java).

Your goal, below, will be to adapt one of those recursive solutions to this puzzle into existing code.

2 Getting started

Get the code: Open the *Terminal*, and at the shell, use the following command to make a directory for this project, download the code, and open it:

```
$ mkdir ~/lab-2
$ cd ~/lab-2
$ curl -L https://bit.ly/cosc-112-24f-l2 -o Towers.java
$ code .
```

Working with the code: You will see that this code does the following:

- Create three arrays of `int` that represent the towers. Each of these tower-arrays are pointed to from an array of `int[]`. That is, an array of three arrays of integers is created.

- The size of each tower (that is, the length of each array of integers) matches the number of rings the puzzle is directed to use by the user.

- Likewise, the integers themselves represent the rings, where a value of `0` indicates no ring (an unoccupied space on the tower), and a positive value indicates a ring of that size. If there are n rings in the puzzle, the rings are sized from 1 to n.

- There is a `print()` method that can print out a textual representation of the towers and the rings on them.

3 Your assignment

Notice that there is a `solve()` method that calls a `doSolve()` method. The first is a *stub method* whose job it is to make the first recursive call to the second. It is the second method (`doSolve()`) that drives the recursive solving. **Your task** is to complete the `doSolve()` method, printing the towers after each move, such that the sequence of moves that solves the puzzle is shown.

Note: You are encouraged to write as many *helper methods* as you need for `doSolve()` to do its work. Break down the task of moving the rings themselves into small methods that call one another.

Another note: Think about how to test this code. Run it with just 1 ring to see if it works correctly. Then 2. Then 3. By the time you work through the 3-disc case, you will likely have found all of your bugs.

4 How to submit your work

Submit your `Towers.java` file by uploading it into the `lab-2` folder in your shared Google Drive folder for this course.

This assignment is due on Sunday, Sep-22, 11:59 pm.