INTRODUCTION TO COMPUTER SCIENCE I
LAB 5
Arrays and randomness

For this lab, you will practice using arrays in a number of different ways. You will also be introduced to *pseudo-random number generators*, and the use of these unpredictable values in your computations.

1 Getting started

Begin like so:

1. **Login** to `remus` or `romulus`.
2. **Open a terminal** to get a shell prompt.
3. **Make a directory** named `lab-5` (using `mkdir`) and change into it (using `cd`).
4. **Open a new source code file for editing** (using `emacs`). The file should be named `ShuffleTest.java`

2 Random numbers

For this assignment, you will need to use the following method that generates *pseudo-random numbers*:

```
double v = Math.random();
```

This method, `Math.random()`, returns a value (here, v) such that $0.0 \leq v < 1.0$. Each time this method is called, it will return a different value. These numbers have two characteristics:

1. They are *uniformly distributed* between 0.0 and not quite 1.0. That is, each value in that range is equally likely to be the next value.
2. They are *unpredictable*. Knowing any number of the values that it previously returned does not help you predict the next value.

For this method, the numbers only *seem* unpredictable. Since we do not know what code is inside this method, we have no good or easy way to predict the next value. However, computers are *deterministic* machines: given the exact same inputs and state, it will always produce the same result. If we could see the source code of `Math.random()`, we could ourselves calculate the next value before calling the method and seeing it.

Therefore, `Math.random()` produces numbers that only seem unpredictable, even though those numbers are, in fact, perfectly predictable with knowledge of the algorithm that produces them. That makes the sequence of numbers that this method produces *pseudo-random*, and not truly random. This distinction will not matter for this lab, but we will discuss the issue later, during lectures.

3 Your assignment

We seek to create a method that can *randomly permute* (that is, *shuffle*) the values in an array. To reorder an array's values, in a (seemingly) random way, the algorithm must obtain and use random numbers. Your code can do so by using the `Math.random()` method described in Section 2.

It will be left to you how to make a method that shuffles an array's contents. There are many ways to do it, and you should not worry about speed or space. That is, if your algorithm is not fast, then that should be acceptable (within reason); likewise, your algorithm may or may not create a new array to contain the shuffled contents.

The following describes the program you must write, `ShuffleTest`, that will test whether your shuffling method works well. The program should:

1. Ask the user for a number of repetitions—that is, the number of times the test described below should be performed.
2. Initialize an array of 100 integers, in which running sums will be kept. Specifically, each position will represent the sum of the values that have landed in that position number in each shuffled array.
3. Repeat the following test for the user-requested number of repetitions:
   (a) Create an array of 100 values. Each entry in the array should be initialized with its own index value. That is, position `[0]` should contain the value `0`, position `[17]` should contain `17`, and so on.
   (b) Pass this array to a method that will shuffle its order.
   (c) For each position i in the shuffled array, add its value to position i in the array of running sums.
4. After performing the tests, print the contents of the sums array. That is, print, one per line, the position number and the value in that position number, over the whole array, like so:

```
0 49672
1 49150
2 48998
3 50012
4 49527
...
99 49463
```

For a good shuffling algorithm, *each value has an equal probability of being placed in each position*. Therefore, over a number of shuffling tests (say, 1,000), each sum in the sums array should contain a number close to the average value of all of the shuffled values, multiplied by the number of tests. That is, with values from 0 to 99 being shuffled, the mean of those values is $\frac{0+99}{2} = 49.5$. If we sum values whose average is that 49.5 over 1,000 iterations, we get $49.5 \times 1,000 = 49,500$.

Therefore, if each of the sums in the 100 positions shown at the end of the program is near that expected sum, then your shuffling code worked well. If there is a skew, with some positions having higher numbers and others lower ones, then your algorithm had a tendency to place larger values in the former positions and smaller values in the latter. In other words, an uneven set of sums implies poor shuffling. Try to make your shuffling method work well.

4 How to submit your work

Use the CS submission systems to submit your work. Specifically, you will need to submit your `ShuffleTest.java` file. You may use either of the following two methods, while connected to `remus` or `romulus`, to use the submission system:

- **Web-based:** Visit the submission system web page.
- **Command-line based:** Use the `~lamcgeoch/submit` command at your shell prompt.

This assignment is due on Thursday, Mar-24, 11:59 pm, before it becomes Friday, Mar-25.