CMSC B113 Computer Science 1 – Spring 2025
Programming Assignment #2

Overview

In this assignment you will write three separate programs (see Part 1, Part 2, and Part 3). Read through all the parts first before starting. You can choose to do the parts in any order.

In completing this assignment, you will learn to:

- Use **if**-statements and **for/while**-loops in a Java program
- Use the **`Math.random()`** function to generate random numbers
- Learn how to implement (code) the steps of an algorithm into Java statements.

Note that in this assignment, you will also be asked to write up a brief report regarding your findings and observations. Be sure to allocate time for doing that write-up, in addition to programming and testing your solution. Please be sure to get an early start on the assignment.

Part 1: Computing the average of N random numbers

Write a program that takes an integer command line argument **`N`**, uses **`Math.random()`** to print N random values in [0.0 .. 1), and then prints their average value.

Let us begin by first writing a program segment to generate and print N random numbers. It should have the following statements:

```
// Input N from the command line
int N = Integer.parseInt(args[0]);

// Generate N random numbers in [0.0 .. 1.0)
for (int i=1; i <= N; i++) {
    double x = Math.random();       // generates a random number
    System.out.println(x);          // prints it out
}
```

The above statements generate N random numbers and print them out. Write a complete Java program called **`Average.java`** (we have not yet done the averaging part, but that will be next) that uses the above statements to generate and print out N random numbers.

Once done, you can run the program as shown below:

```
$ java-introcs Average 10
```

It should print out ten random numbers in the range [0.0 .. 1.0). Observe the numbers to ensure that they do appear random. Run the program again to generate ten more random numbers. Did you get the same ten random numbers? They should be different. That is, **`Math.random()`** generates different sets of ransom numbers each time it is used.

Next, to compute the average of N numbers, here is the algorithm:

```
1. Input N
2. Set sum ← 0.0
3. Repeat N times
   3.1 Set x ← random number in [0.0 .. 1.0)
   3.3 Print x
   3.4 Add x to sum
4. Set average ← sum/N
5. Output the average
```

Modify **`Average.java`** to implement the above algorithm. Run it for N = 10 several times and note the average value printed. Is it always the same? It shouldn't be. Is it close to 0.5?

If the numbers are random, their average should be 0.5. Right? This is true only for very large values of N. Before you try to run your program for very large values of N, comment out the code that implements Step 3.3, so you will not print screen and screen of random numbers!

Next, run the program for N = 100, 1000, 10000, etc. (you can go as high as 1 billion!) until you see values very close to 0.5 (say within +/- 0.00001. What is that N?

<u>What to hand in:</u>

For **`Average.java`**, print out the final version of the complete program. Run the program for N = 10, showing the random numbers generated and their average. Next, run the program for a value of N that converges to within 0.5 +/- 0.00001. Include these outputs at the end of your program. You may cut and paste the program runs and their outputs.

<u>Part 2: Computing Square Root of a number</u>

Let us learn how to compute square roots of a number using a calculator. Then you will write a Java program to do the task.

To compute the square root $x = \sqrt{a}$ do the following:

1. Start with some guess $x_1 > 0$
2. Compute a sequence of guesses $x_1, x_2, \ldots, x_n$ using the equation:

$$x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$$

until the numbers produced converge.

Do This: Put down this book and get a piece of scratch paper and pencil. And a calculator. You will need one. Compute the square root of *a* = *2* (i.e. $\sqrt{2}$) using the above formula. Start with $x_1 = 1$. Next, write down your answer for values of x_1, x_2, x_3, etc. You should get something like the sequence shown below:

$x_1 = 1$
$x_2 = 1.5$
$x_3 = 1.416666667$
$x_4 = 1.414215686$
$x_5 = 1.414213563$
$x_6 = 1.414213562$
$x_7 = 1.414213562$

Were you also able to compute $\sqrt{2}$ correctly?

Depending on the calculator you used you will get the same result as above. On our calculator, we were able to get values with a precision of 9-digits after the decimal. Yours may be different in the number of digits of accuracy. We stopped after the same value was produced twice in a row (see x_6 and x_7 above). That is, the sequence *converged*.

Congratulations! You were just a computer. And, you did some computing!

Next, we will use the algorithm above to compute square roots on the computer. Write a Java program **`Sqrt.java`** that inputs a number ***a*** (type double) and computes and prints out $\sqrt{a}$. For example,

```
$ java-introcs Sqrt 2.0
The square root of 2.0 is 1.414213562373095
```

```
$ java-introcs Sqrt 3.0
The square root of 3.0 is 1.732058075688772
```

While the steps to compute the square root we presented above were sufficient for us to follow and successfully compute $\sqrt{a}$ we will need to be a little more precise in specifying the algorithm into steps that can be coded into a computer program. Here is such an algorithm:

```
1. Input *a*
2. Set $x_0 \leftarrow 0.0$
3. Set $x_1 \leftarrow 1.0$
4. While $x_1 \neq x_0$ do
4.1 Set $x_0 \leftarrow x_1$
4.2 Set $x_1 \leftarrow \frac{1}{2}(x_0 + \frac{a}{x_0})$
5. Output $x_1$
```

Notice that we translated x_1, x_2, … to just two variables x_0 and x_1. You can use the variable names X0 and X1 isn your Java program since Java does not allow subscripts.

Study the algorithm carefully, and write down the Java commands that implement each of the steps in your notebook. Then enter the program, compile, and run it to ensure correctness.

What to Hand in

Once completed, as you did in Part 1, print out the final version of the complete program. Run the program for inputs 2.0 and 3.0, showing the results Include these outputs at the end of your program. You may cut and paste the program runs and their outputs.

Part 3: Figure Skating Judging

In the winter Olympics, a figure skater's score is determined by a panel of six judges who each decide a score between 0.0 and 10.0. The final score is determined by discarding the high and low scores and averaging the remaining 4. Write a program `SkatingScorer.java` that takes 6 real command line inputs representing the 6 scores and prints their average, after throwing out the high and low scores.

Start, as above, by first writing an algorithm on a piece of paper. In fact, focus first on computing the average of all six scores. Once that is completed, add the steps to compute the smallest score. Here is an algorithm to compute the smallest of three numbers:

To compute the smallest of three numbers: *a*, *b*, and *c* do the following:

```
1. Assume that a is the smallest number: Set $min \leftarrow a$
2. See if b is smaller than min: if $b < min$
2.1. Set $min \leftarrow b$
3. See if $c$ is smaller than min: if $c < min$
3.1 Set $min \leftarrow c$
4. $min$ now contains the smallest number in $a$, $b$, and $c$.
```

You can extend the above for six numbers. Write and test your program to compute the smallest of the six numbers and eliminate it from the average. Once done, test the program to see that you are getting correct results. Finally, adapt the above to find the largest score and eliminate from computing the average.

What to Hand in

Once completed, as you did in Parts 1 and 2, print out the final version of the complete program. Run the program for inputs [9.0, 9,5, 10.0, 8.2, 9.0, 9.1] and [10.0, 10.0, 10.0, 9.0, 10.0, 10.0], and [8.0, 8.0, 8.5, 9.0, 8.5, 9.5], showing the results. Include these outputs at the end of your program. You may cut and paste the program runs and their outputs.

What/How to Submit

Once you have completed the assignment, write a short reflection on your experience and what you learned from this assignment. To print all the parts, you can create a *single* text document, and print the document as your submission.