CIBC Java Quant Dev Coding Quiz

Instructions

1. You agree not to disclose the quiz.
2. You are not allowed to discuss the questions with anyone.
3. Please provide a Java implementation for each problem. Submissions without an implementation for all problems will not be considered.
4. The performance of each solution matters: consider how to minimise both the execution time and memory usage.
5. Provide any documentation, test cases, or any other material that you consider relevant to support the correctness of your solutions. Please do not submit binaries.
6. Where possible submit all materials in a single .7z file. Please do not use any other archive format.
7. The focus is on the methodology and the ideas behind the design, plus some basic understanding in core concepts related to the server side development in high frequency low latency trading applications.

Algorithms1

Implement the method nextNum() and a minimal but effective set of unit tests. Implement in Java. Make sure your code is exemplary, as if it was going to be shipped as part of a production system.

As a quick check, given Random Numbers are [-1, 0, 1, 2, 3] and Probabilities are [0.01, 0.3, 0.58, 0.1, 0.01] if we call nextNum() 100 times
we may get the following results. As the results are random, these particular results are unlikely.
-1: 1 times
0: 22 times
1: 57 times
2: 20 times
3: 0 times

You may use Random.nextFloat() which returns a pseudo random number between 0 and 1.

```
import org.apache.commons.lang.math.RandomUtils;

public class RandomGen {
    final private int[] randomNums;
    final private float[] probabilities;

    public RandomGen(final int[] randomNums, final float[] probabilities) {
        this.randomNums = randomNums;
        this.probabilities = probabilities;
        init();
    }

    private void init() {
        if (randomNums.length != probabilities.length) {
            throw new IllegalArgumentException("The array length are not consistent");
        }
        if (randomNums.length == 0) {
            throw new IllegalArgumentException("The array length is not legal");
        }
        // add initialisation procedure below to deliver the most efficient solution
```

```
    }

    /**
    Returns one of the randomNums. When this method is called
    multiple times over a long period, it should return the
    numbers roughly with the initialized probabilities.
    */
    public int nextNum() {

    }

}
```

Algorithms1

The king pays his loyal knight in gold coins. On the first day of his service, the knight receives one gold coin. On each of the next two days (the second and third days of service), the knight receives two gold coins. On each of the next three days (the fourth, fifth, and sixth days of service), the knight receives three gold coins. On each of the next four days (the seventh, eighth, ninth, and tenth days of service), the knight receives four gold coins. This pattern of payments will continue indefinitely: after receiving N gold coins on each of N consecutive days, the knight will receive N+1 gold coins on each of the next N+1 consecutive days, where N is any positive integer.
Your program will determine the total number of gold coins paid to the knight in any given number of days (starting from Day 1).

Input

Each line of the input (except the last one) contains data for one test case of the problem, consisting of exactly one integer (in the range 1..10000), representing the number of days. The end of the input is signaled by a line containing the number 0.

Output

There is exactly one line of output for each test case. This line contains the number of days from the corresponding line of input, followed by one blank space and the total number of gold coins paid to the knight in the given number of days, starting with Day 1.

Sample Input Sample Output

10 -> 10 30

6 -> 6 14

7 -> 7 18

11 -> 11 35

15 -> 15 55

16 -> 16 61

100 -> 100 945

10000 -> 10000 942820

1000 -> 1000 29820

21 -> 21 91

22 -> 22 98

0 //end