Hailstone

1. SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

A mathematical sequence is an ordered list of numbers. This question involves a sequence called a *hailstone sequence*. If n is the value of a term in the sequence, then the following rules are used to find the next term, if one exists.

- If n is 1, the sequence terminates.
- If n is even, then the next term is $\frac{n}{2}$.
- If n is odd, then the next term is $3n + 1$.

For this question, assume that when the rules are applied, the sequence will eventually terminate with the term $n = 1$.

The following are examples of hailstone sequences.

Example 1: 5, 16, 8, 4, 2, 1

- The first term is 5, so the second term is $5 * 3 + 1 = 16$.
- The second term is 16, so the third term is $\frac{16}{2} = 8$.
- The third term is 8, so the fourth term is $\frac{8}{2} = 4$.
- The fourth term is 4, so the fifth term is $\frac{4}{2} = 2$.
- The fifth term is 2, so the sixth term is $\frac{2}{2} = 1$.
- Since the sixth term is 1, the sequence terminates.

Example 2: 8, 4, 2, 1

- The first term is 8, so the second term is $\frac{8}{2} = 4$.
- The second term is 4, so the third term is $\frac{4}{2} = 2$.
- The third term is 2, so the fourth term is $\frac{2}{2} = 1$.
- Since the fourth term is 1, the sequence terminates.

The `Hailstone` class, shown below, is used to represent a hailstone sequence. You will write three methods in the `Hailstone` class.

```
public class Hailstone
{
     /** Returns the length of a hailstone sequence that starts with n,
       * as described in part (a).
       * Precondition: n > 0
       */
     public static int hailstoneLength(int n)
```

Hailstone

```
    { /* to be implemented in part (a) */ }
  /** Returns true if the hailstone sequence that starts with n is
considered long
      * and false otherwise, as described in part (b).
      * Precondition: n > 0
      */
    public static boolean isLongSeq(int n)
    { /* to be implemented in part (b) */ }
  /** Returns the proportion of the first n hailstone sequences that are
considered long,
      * as described in part (c).
      * Precondition: n > 0
      */
    public static double propLong(int n)
    { /* to be implemented in part (c) */ }
  // There may be instance variables, constructors, and methods not
shown.
}
```

(a) The length of a hailstone sequence is the number of terms it contains. For example, the hailstone sequence in example 1 (5, 16, 8, 4, 2, 1) has a length of 6 and the hailstone sequence in example 2 (8, 4, 2, 1) has a length of 4.

Write the method `hailstoneLength(int n)`, which returns the length of the hailstone sequence that starts with `n`.

```
/** Returns the length of a hailstone sequence that starts with n,
 * as described in part (a).
 * Precondition: n > 0
 */
public static int hailstoneLength(int n)
```

Class information for this question

```
public class Hailstone
public static int hailstoneLength(int n)
public static boolean isLongSeq(int n)
public static double propLong(int n)
```

(b) A hailstone sequence is considered long if its length is greater than its starting value. For example, the hailstone sequence in example 1 (5, 16, 8, 4, 2, 1) is considered long because its length (6) is greater than its starting value (5). The hailstone sequence in example 2 (8, 4, 2, 1) is not considered long because its length (4) is less than or equal to its starting value (8).

Hailstone

Write the method `isLongSeq(int n)`, which returns `true` if the hailstone sequence starting with `n` is considered long and returns `false` otherwise. Assume that `hailstoneLength` works as intended, regardless of what you wrote in part (a). You must use `hailstoneLength` appropriately to receive full credit.

```
/** Returns true if the hailstone sequence that starts with n is
considered long
 * and false otherwise, as described in part (b).
 * Precondition: n > 0
 */
public static boolean isLongSeq(int n)
```

(c) The method `propLong(int n)` returns the proportion of long hailstone sequences with starting values between 1 and `n`, inclusive.

Consider the following table, which provides data about the hailstone sequences with starting values between 1 and 10, inclusive.

Starting Value	Terms in the Sequence	Length of the Sequence	Long?
1	1	1	No
2	2, 1	2	No
3	3, 10, 5, 16, 8, 4, 2, 1	8	Yes
4	4, 2, 1	3	No
5	5, 16, 8, 4, 2, 1	6	Yes
6	6, 3, 10, 5, 16, 8, 4, 2, 1	9	Yes
7	7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1	17	Yes
8	8, 4, 2, 1	4	No
9	9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1	20	Yes
10	10, 5, 16, 8, 4, 2, 1	7	No

The method call `Hailstone.propLong(10)` returns `0.5`, since 5 of the 10 hailstone sequences shown in the table are considered long.

Write the `propLong` method. Assume that `hailstoneLength` and `isLongSeq` work as intended, regardless of what you wrote in parts (a) and (b). You must use `isLongSeq` appropriately to receive full credit.

```
/** Returns the proportion of the first n hailstone sequences that are
considered long,
```

Hailstone

```
 * as described in part (c).
 * Precondition: n > 0
 */
public static double propLong(int n)
```

Class information for this question

```
public class Hailstone
public static int hailstoneLength(int n)
public static boolean isLongSeq(int n)
public static double propLong(int n)
```