Chapter 10 Recursive Programming Project (Do all problems in one class: Each problem should be its own method)

#1)

Write a recursive method called `starString` that accepts an integer as a parameter and prints to the console a string of stars (asterisks) that is 2^n (i.e., 2 to the n^{th} power) long. For example,

- `starString(0)` should print `*` (because $2^0 == 1$)
- `starString(1)` should print `**` (because $2^1 == 2$)
- `starString(2)` should print `****` (because $2^2 == 4$)
- `starString(3)` should print `********` (because $2^3 == 8$)
- `starString(4)` should print `****************` (because $2^4 == 16$)

The method should throw an `IllegalArgumentException` if passed a value less than `0`.

#2)

Write a method called `writeNums` that takes an integer `n` as a parameter and prints to the console the first `n` integers starting with 1 in sequential order, separated by commas. For example, consider the following calls:

```
writeNums(5);
System.out.println(); // to complete the line of output
writeNums(12);
System.out.println(); // to complete the line of output
```

These calls should produce the following output:

```
1, 2, 3, 4, 5
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
```

Your method should throw an `IllegalArgumentException` if passed a value less than `1`.

#3)

Write a recursive method called `sumTo` that accepts an integer parameter *n* and returns a real number representing the sum of the first *n* reciprocals. In other words, `sumTo(n)` returns (1 + 1/2 + 1/3 + 1/4 + ... + 1/*n*). For example, `sumTo(2)` should return `1.5`. The method should return `0.0` if it is passed the value 0 and throw an `IllegalArgumentException` if it is passed a value less than 0.

#4)

Write a recursive method called `repeat` that accepts a string *s* and an integer *n* as parameters and that returns *s* concatenated together *n* times. For example, `repeat("hello", 3)` returns `"hellohellohello"`, and `repeat("ok", 1)` returns `"ok"`, and `repeat("bye", 0)` returns `""`. String concatenation is an expensive operation, so for an added challenge try to solve this problem while performing fewer than *n* concatenations.

#5)

Write a recursive method called `isReverse` that accepts two strings as parameters and returns `true` if the two strings contain the same sequence of characters as each other but in the opposite order (ignoring capitalization), and `false` otherwise. For example, the call of `isReverse("hello", "eLLoH")` would return `true`. The empty string, as well as any one-letter string, is considered to be its own reverse.