Comp 271 - Assignment
Towers of Hanoi with Metrics
As in the course syllabus, late assignments will not be accepted unless you have completed the process for using a late pass before the due date.

1. Submit
The zipped project file in Sakai assignment following the correct file naming convention.

2. Format
Java program using proper programming style and with appropriate java doc comments in the code. Code must compile and run in the tools used in class.

3. Content
Summary: Write a complete Java project using correct programming techniques and style. The purpose of this assignment work with recursion and get a feel for how efficient it is as a problem solving technique.

Your program is to

1. Implement a recursive solution to Towers of Hanoi, derived from the text example case study, Chapter 5, starting on page 238, with some variations.
2. You are required to use recursion; you can start from the book solution and make modifications as you wish. This assignment does not demand Object Oriented solutions, but you are encouraged to think about how and if to add custom classes.
3. You are required to do Junit testing of your showMoves method. You pick which tests to do and how to implement them. You can test other methods too if you add some or have other classes.
4. Add the number of the move to each line of the output the text solution generates. Line one becomes “1: Move disk 1 from peg L to peg R”
5. Modify the text solution so that showMoves takes an additional (5th) parameter, a Boolean that turns showing individual moves on (value true) or off (value false).
6. In all cases (independent of showing moves in the return value) you need to include these two metrics which you need to calculate during the recursion (do not use any closed form to calculate the answer)
   - What is the total number of disk moves made?
   - How many times has showMoves been called?
   - Include both these metrics in the final line (or only line if individual move display suspended) displayed using a format similar to “To solve *n* disks, called function *y* times and moved a disk *z* times”.
   - Format y and z in a pleasant and readable way such as using commas.

7. Test your program with at least 1, 2, 3, 4, 10, and 15 disks. You can do more test cases. Include these in your Junit testing. You decide how to compare expected and actual results.

8. Include in comments in your main program your conclusion as to the Big O for your code.

Test your program fully and carefully. Make sure the output from the program looks good and is easy to follow.

4. Hints and Background

1. First make sure you can get the text answer for 3 disks! You probably want to turn off the individual move generation for larger numbers.
2. Any reason you need to add a wrapper method?
3. Legend holds that somewhere in the high Himalayas there are a group of monks who work to complete the puzzle with 64 thin gold disks. The reason that you cannot put a larger disk on the smaller one is that the gold would easily bend if a heavier disk were placed on top of a smaller one. The monks started during the time when The Buddha walked the earth, have been working ever since, and the task is passed done to new generations of monks when older ones reincarnate. When they finish moving all disks the world will end! (Don't try this case in your program unless you are very patient).

5. Grading

20 points total. No points if will not compile in the tools used in class. -5 to -10 for logic errors such as incorrect metrics, making incorrect moves, etc. depending on severity. -10 for no Junit testing. Up to -10 for poor programming style or lack of documentation. -5 for not giving a Big O answer in your comments in main.

TowersOfHanoiWithMetrics v2
Version 2 (clarify last line metrics and options for creating)
Version 1 (java edition) Mar 2017
©W.L.Honig for Comp 271 Loyola University Chicago, Spring 2017
//end assignment