Fibonacci: Recursion vs. Iteration

1. Complete the Fibonacci methods below. Write TWO answers for this problem: one using recursion and one using iteration.

/\*\*\*\*\*\*\*\*\*\*

pre-condition: *n* > 0

post-condition: returns the *n*th Fibonacci #: 1, 1, 2, 3, 5, 8, 13,..

\*\*\*\*\*\*\*\*\*\*/

| public static long fibRecur(int n)  { | public static long fibIter(int n)  { |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| } | } |

| 2. Trace the recursive calls for fibRecur(5). | 3. Which do you think will be quicker, Fibonacci by recursion or Fibonacci by iteration? Why? |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

4. Complete the shell and record both run-times for the given Fibonacci numbers. Graph both curves on the graph paper on the back. What kinds of curves are they? Explain why the results make sense.

5. Complete the hailstone methods below. Use recursion. Each is recursive, but counts the steps in different ways. For the input 12, output is: 12-6-3-10-5-16-8-4-2-1 takes 10 steps

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

pre-condition: n > 0

post-condition: prints the hailstone sequence that starts with n.

Counts and returns the number of steps from n to 1.  
If n is even then the next number is n / 2.

If n is odd then the next number is 3 \* n + 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

| //recursive, prints the sequence, counts the // steps with a variable  public static int hailstone(int n, int count)  { | //recursive, print the sequence, //counts the steps without a variable  public static int hailstone(int n)  { |
| --- | --- |

|  |  |
| --- | --- |

