Warm-up questions

1. What are the two principal characteristics of a recursive algorithm?
    1. Each recursive call should be on a smaller instance of the same problem.
    2. The recursive call must eventually reach a base case, which is solved without further recursion.

2. Recursion is..

Answer	
	theoretically interesting but rarely used in actual programs
	theoretically uninteresting and rarely used in programs
X	theoretically powerful and often used in algorithms that could benefit from recursive methods

3. True or false: All recursive functions can be implemented iteratively

True

4. True or false: if a recursive algorithm does NOT have a base case, the compiler will detect this and throw a compile error?

False

5. True or false: a recursive function must have a void return type.

False

6. True or False: Recursive calls are usually contained within a loop.

False

7. True or False: Infinite recursion can occur when a recursive algorithm does not contain a base case.

True

8. Which of these statements is true about the following code?

```
int mystery(int n)
{
        if (n>0) return n + mystery(n-1);
        return 0;
}
```

Your answer	

	The base case for this recursive method is an argument with any value which is greater than zero.
X	The base case for this recursive function is an argument with the value zero.
	There is no base case.

9. List common bugs associated with recursion?

Missing a base case
The sub-problem is not smaller than the original problem i.e. the recursive call isn't being made with a smaller argument each time

10. What method can be used to address recursive algorithms that excessively recompute?

Memoisation can be used to address recursive algorithms that excessively recompute. This method speeds up alogrithms by keeping track of expensive operations and returning the cached result if the same computation is required again. It uses a lookup table.

Fibonacci

The Fibonacci numbers are a sequence of integers in which the first two elements are 0 and 1, and each following element is the sum of the two preceding elements:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, and so on…

The Nth Fibonacci number is output with the following function:

$$fib(n) = fib(n-1) + fib(n-2) \rightarrow \text{for } n > 1$$
$$fib(n) = 1 \rightarrow \text{for } n = 0, 1$$

The first two terms of the series are 0, 1.
For example: fib(0) = 0, fib(1) = 1, fib(2) = 1

Exercises

1. Below is an iterative algorithm that computes Fibonacci numbers. Write a recursive function to do the same.
2. Test both algorithms with various sizes of Ns. What do you find?
   The methods don't return the same answer – the recursive one returns one ahead in the sequence.
3. What is the time complexity of both functions?

Iterative one is O(N) and the recursive one is O(2^n)

Iterative Fibonacci

```
static int fibonacciIterative(int n){
  if (n<=1)
      return 1;

  int fib = 1;
  int prevFib =  1;

  for (int i = 2; i < n; i++) {
   int temp = fib;
   fib = fib + prevFib;
   prevFib = temp;
  }
  return fib;
 }


 public static void main (String args[])
    {
    int n = 9;
    System.out.println(fibonacciIterative(n));
    }
```