Warm-up questions

1. What are the two principal characteristics of a recursive algorithm?

1.Every recursive definition must have one (or more) base cases

2.The general case must eventually reduce to a base case.

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

False

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.

	The base case for this recursive function is an argument with the value zero.
X	There is no base case.

9. List common bugs associated with recursion?

No base case
If the problem requires multiple bases cases
General case never reaches the base case.

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

Memorization

1. Below is an iterative algorithm that computes Fibonacci numbers. Write a recursive function to do the same.

```
1
2 public class FibonacciRecursive {
3
4 static int fibonacciRecursive(int n){
5
6     if(n <= 1) {
7         return n;
8     }
9     else {
10        return fibonacciRecursive(n-1) + fibonacciRecursive(n-2);
11    }
12 }
13
14 public static void main(String[] args) {
15
16    System.out.println(fibonacciRecursive(15));
17    }
18 }
19
```

2. Test both algorithms with various sizes of Ns. What do you find?

Both get the same answer each time.

	N = 10	N = 25
Iterative	0.0	1.0
Recursive	1.0	2.0

*The java class must be rounding these results.

3. What is the time complexity of both functions?

Iterative: O(n)

Recursive O(2^n)

Towers Of Hanoi

I couldn't get the pseudo code from the practical sheet to work correctly: (2 disks)

```
4    static void towerOfHanoi(int n, char source, char auxiliary, char dest){
5          if (n == 0)
6            {
7              System.out.println("Move disk from " + source + " to " + dest);
8            }
9          else {
10           towerOfHanoi(n - 1, source, auxiliary, dest);
11           towerOfHanoi(n - 1, auxiliary, dest, source);
12         }
13   }
```

Tasks Console ×

<terminated> TowerOfHanoi [Java Application] C:\Users\CianO\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.

```
Move disk from A to C
Move disk from B to A
Move disk from B to A
Move disk from C to B
```

However I found a version on tutorialspoint.com that works and it makes sense. (3 disks)

```
4    static void towerOfHanoi(int n, char source, char aux, char dest) {
5          if (n == 1) {
6              System.out.println("Disk 1 from " + source + " to " + dest);
7            } else {
8                towerOfHanoi(n - 1, source, dest, aux);
9              System.out.println("Disk " + n + " from " + source + " to " + dest);
10             towerOfHanoi(n - 1, aux, source, dest);
11           }
12   }
13
```

Tasks Console ×

<terminated> TowerOfHanoi [Java Application] C:\Users\CianO\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64

```
Disk 1 from A to C
Disk 2 from A to B
Disk 1 from C to B
Disk 3 from A to C
Disk 1 from B to A
Disk 2 from B to C
Disk 1 from A to C
```

Link to the code I got from tutorials point

https://www.tutorialspoint.com/javaexamples/method_tower.htm