


Algorithms Practical 3

Gearoid Mulligan: 19343146

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

The two principal characteristics for a recursive algorithm is a general case and a recursive case.

2. Recursion is..

Answer	
	theoretically interesting but rarely used in actual programs
	theoretically uninteresting and rarely used in programs
	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.

True

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?

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.
	There is no base case.

9. List common bugs associated with recursion?

Missing a base case	function will repeatedly call itself & never return
Excessive Re-computation	sometimes a seemingly simple recursive program can require exponential time
The recursive step doesn't reduce to a smaller subproblem	the recursion doesn't converge
the problem needs more than one base case but not all the base cases are covered	function will repeatedly call itself & never return

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

We can use a technique called memoization to save the computer time when making identical function calls. Memoization (a form of caching) remembers the result of a function call with particular inputs in a lookup table (the "memo") and returns that result when the function is called again with the same inputs.

Fibonacci

2. I realised that as the size of n increased, it took the recursive function a lot longer than the iterative function to figure out the correct value

3. For the recursive Fibonacci is $O(2^n)$ (exponential).

For the iterative Fibonacci it is $O(n)$ as the code is linear.