

Practical 3: Recursion

What am I doing today?

Today's practical focuses on 3 things:

1. Quick questions about Recursion
2. Comparing an iterative fibonacci algorithm to a recursive one
3. Help the monks solve the Towers of Hanoi

Instructions

Try all the questions. Ask for help from the demonstrators if you get stuck.

Solutions will be posted afterward.

*****Grading: Remember** if you complete the practical, add the code to your GitHub repo which needs to be submitted at the end of the course **for an extra 5%**

Student Name = Adam Leacy

Student Number = 18313471

Warm-up questions

1. What are the two principal characteristics of a recursive algorithm?
 - A recursive algorithm must have a base-case
 - A recursive algorithm must call itself

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.
TRUE
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	There is no base case
	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	The function will repeatedly call itself & never return
No guarantee of convergence	The sub-problem is not smaller than the original problem
Excessive memory requirements	Excessive self-calls before returning, memory Java needs to keep track may be too much
Excessive re-computation	Some times a seemingly simple recursive program can require exponential time

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

Memorisation

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:

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

$$\text{fib}(n) = 1 \rightarrow \text{for } n = 0, 1$$

The first two terms of the series are 0, 1.

For example: $\text{fib}(0) = 0$, $\text{fib}(1) = 1$, $\text{fib}(2) = 1$

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?
Larger sizes of N take longer to compute when using the recursive function
3. What is the time complexity of both functions?
Recursive functions can take longer to calculate a final result

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));
}
```


Hanoi - The Monks need your help!



Convert the pseudo-code into java and add your own output instructions so junior monks can learn how to perform the legal moves in the Tower of Hanoi so they can end the world.

There are two rules:

- Move only one disc at a time.
- Never place a larger disc on a smaller one.

Tasks:

1. Implement Hanoi in java
2. Test with various size disks
3. Output the moves for the monks as step-by-step instructions so the monks can end the world

Pseudocode for Hanoi

```
towersOfHanoi(disk, source, dest, auxiliary):  
IF n == 0, THEN:  
    move disk from source to dest  
ELSE:  
    towersOfHanoi(disk - 1, source, auxiliary, dest)  
    towersOfHanoi(disk - 1, auxiliary, dest, source)
```

END IF