



[Course](#) [Progress](#)

[Course](#) > [Readings](#) > [Reading 1: Recursion](#) > [Questions](#)


[Previous](#)

































[Next](#)

## Questions

 [Bookmark this page](#)

### Recursive factorial

1/1 point (graded)

Consider this recursive implementation of the factorial function.

```
public static int factorial(int n) {  
    if (n == 0) {  
        return 1; // this is called the base case  
    } else {  
        return n * factorial(n-1); // this is the recursive step  
    }  
}
```

For factorial(3), how many times will the base case be executed?

☐ 0 times

☒ 1 time

☐ 2 times

☐ 3 times


☐ more than 3 times



#### Explanation

factorial(3) calls factorial(2), which calls factorial(1), which calls factorial(0), which executes the base case once. Each of the recursive calls then returns, without any further recursion.

[Submit](#)

 [Show Answer](#)

 Answers are displayed within the problem

### Recursive Fibonacci

1/1 point (graded)

Consider this recursive implementation of the Fibonacci sequence.

```
public static int fibonacci(int n) {  
    if (n == 0 || n == 1) {  
        return 1; // base cases  
    } else {  
        return fibonacci(n-1) + fibonacci(n-2); // recursive step  
    }  
}
```

For fibonacci(3), how many times will the base case be executed?

☐ 0 times

☐ 1 time

☐ 2 times

☒ 3 times

☐ more than 3 times



#### Explanation

fibonacci(3) calls fibonacci(2) which calls fibonacci(1), which executes the base case once, and fibonacci(0) which executes the base case again. fibonacci(2) then returns to fibonacci(3), which now calls fibonacci(1), which executes the base case one more time, for a total of 3.

Another way to do it: because fibonacci is a pure function, with no side-effects, we can substitute expressions:

$$\begin{aligned}\text{fibonacci}(3) &= \text{fibonacci}(2) + \text{fibonacci}(1) \\ &= (\text{fibonacci}(1) + \text{fibonacci}(0)) + \text{fibonacci}(1) \\ &= (1 + 1) + 1\end{aligned}$$

Still another way to check, which only works for Fibonacci, is to realize that each execution of the base case contributes 1 to the final answer, and none of the recursive steps contribute any additional value. So the number of base case executions in fibonacci(n) is the same as the nth Fibonacci number.

Submit

Show Answer

Answers are displayed within the problem

[< Previous](#)

[Next >](#)

Some Rights Reserved

[Open Learning Library](#)

[About](#)

[Accessibility](#)

[All Courses](#)

[Why Support MIT Open Learning?](#)

[Help](#)

[Connect](#)

[Contact](#)

[Twitter](#)

[Facebook](#)

[Privacy Policy](#) [Terms of Service](#)

© Massachusetts Institute of Technology, 2024