QUIZ

Fill in the `.findNodeRecursively()` method, so it returns the correct node when found, or `null` if not found.

```
findNodeRecursively( data , currentNode = this.head ) {
    if (currentNode === null) {
      return null ;
    } else if (currentNode.data === data) {
      return currentNode ;
    } else {
      return this.findNodeRecursively (data, currentNode.next );
    }
  }
```

You got it!

Fill in the code, so `recursiveSum()` recursively finds the sum of a number and all positive numbers less than it.

```
const recursiveSum = ( n ) => {
  if ( n === 1 ) {
    return 1 ;
  }

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

You got it!

Fill in the correct conditions for the `.findNodeRecursively()` method.

```
findNodeRecursively(data, currentNode = this.head) {
    if ( currentNode === null ) {
      return null;
    } else if ( currentNode.data === data ) {
      return currentNode;
    } else {
      return this.findNodeRecursively(data, currentNode.next);
    }
  }
```

You got it!

The function `recursiveFactorial()` uses recursion to find the factorial solution to the argument passed into it. Which of the following is a realistic call stack for the call `recursiveFactorial(3)`?

1. `recursiveFactorial(2) = 2 * recursiveFactorial(1)`
2. `recursiveFactorial(1) = 1 * recursiveFactorial(0)`
3. `recursiveFactorial(3) = 3 * recursiveFactorial(2)`

1. `recursiveFactorial(1) = 1 * recursiveFactorial(0)`
2. `recursiveFactorial(2) = 2 * recursiveFactorial(1)`
3. `recursiveFactorial(3) = 3 * recursiveFactorial(2)`

1. `recursiveFactorial(3) = 3 * recursiveFactorial(2)`
2. `recursiveFactorial(2) = 2 * recursiveFactorial(1)`
3. `recursiveFactorial(1) = 1 * recursiveFactorial(0)`

Nice work! The `recursiveFactorial()` function will call itself with inputs of `0`, `1`, `2`, and `3` and multiply the solution by `n+1` values. The top of the stack is the first call.

The code displays a recursive solution to finding a node in a linked list. Which of the following is the condition for the recursive case?

```
findNodeRecursively(data, currentNode = this.head) {
    if (currentNode === null) {
      return null;
    } else if (currentNode.data === data) {
      return currentNode;
    } else {
      return this.findNodeRecursively(data, currentNode.next);
    }
  }
```

The recursive condition is if `currentNode.data === data` and `currentNode === null` are not `true`

👏 Nice work! The recursive case executes if the first two conditions are not true.

`currentNode.data === data`

`currentNode === null`

Which of the following errors often occurs if you do not include a base case in your recursive function?

Key error

Import error

A stack overflow

👏 Nice work! If the function does not have a base case, then it may continue to call itself indefinitely.

A value error

Fill in the code, so `recursiveFactorial()` recursively finds the factorial solution to an argument and returns it.

```
const recursiveFactorial = ( n ) => {
  if ( n === 0 ) {
    return 1 ;
  }
  if ( n > 0 ){
    return recursiveFactorial ( n - 1 ) * n;
  }
}
```

You got it!