# Problem 2648: Generate Fibonacci Sequence

## Problem Information

**Difficulty:** Easy
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

Write a generator function that returns a generator object which yields the

fibonacci sequence

.

The

fibonacci sequence

is defined by the relation

X

n

= X

n-1

+ X

n-2

.

The first few numbers of the series are

0, 1, 1, 2, 3, 5, 8, 13

.

Example 1:

Input:

callCount = 5

Output:

[0,1,1,2,3]

Explanation:

const gen = fibGenerator(); gen.next().value; // 0 gen.next().value; // 1 gen.next().value; // 1 gen.next().value; // 2 gen.next().value; // 3

Example 2:

Input:

callCount = 0

Output:

[]

Explanation:

gen.next() is never called so nothing is outputted

Constraints:

0 <= callCount <= 50

## Code Snippets

### JavaScript:

```
/**
* @return {Generator<number>}
*/
var fibGenerator = function*() {

};

/**
* const gen = fibGenerator();
* gen.next().value; // 0
* gen.next().value; // 1
*/
```

### TypeScript:

```
function* fibGenerator(): Generator<number, any, number> {

};

/**
* const gen = fibGenerator();
* gen.next().value; // 0
* gen.next().value; // 1
*/
```

## Solutions

### JavaScript Solution:

```
/**
* Problem: Generate Fibonacci Sequence
* Difficulty: Easy
* Tags: general
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
```

```
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @return {Generator<number>}
*/
var fibGenerator = function*() {

};

/**
* const gen = fibGenerator();
* gen.next().value; // 0
* gen.next().value; // 1
*/
```

**TypeScript Solution:**

```
/**
* Problem: Generate Fibonacci Sequence
* Difficulty: Easy
* Tags: general
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function* fibGenerator(): Generator<number, any, number> {

};

/**
* const gen = fibGenerator();
* gen.next().value; // 0
* gen.next().value; // 1
*/
```