Problem 2665: Counter II

Problem Information

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

Problem Description

Write a function

createCounter

. It should accept an initial integer

init

. It should return an object with three functions.

The three functions are:

increment()

increases the current value by 1 and then returns it.

decrement()

reduces the current value by 1 and then returns it.

reset()

sets the current value to

init

and then returns it.

Example 1:

Input:

init = 5, calls = ["increment","reset","decrement"]

Output:

[6,5,4]

Explanation:

const counter = createCounter(5); counter.increment(); // 6 counter.reset(); // 5
counter.decrement(); // 4

Example 2:

Input:

init = 0, calls = ["increment","increment","decrement","reset","reset"]

Output:

[1,2,1,0,0]

Explanation:

const counter = createCounter(0); counter.increment(); // 1 counter.increment(); // 2
counter.decrement(); // 1 counter.reset(); // 0 counter.reset(); // 0

Constraints:

-1000 <= init <= 1000

0 <= calls.length <= 1000

calls[i]

is one of "increment", "decrement" and "reset"

Code Snippets

JavaScript:

```
/**
* @param {integer} init
* @return { increment: Function, decrement: Function, reset: Function }
*/
var createCounter = function(init) {

};

/**
* const counter = createCounter(5)
* counter.increment(); // 6
* counter.reset(); // 5
* counter.decrement(); // 4
*/
```

TypeScript:

```
type Counter = {
increment: () => number,
decrement: () => number,
reset: () => number,
}

function createCounter(init: number): Counter {

};

/**
* const counter = createCounter(5)
* counter.increment(); // 6
* counter.reset(); // 5
* counter.decrement(); // 4
*/
```

Solutions

JavaScript Solution:

```
/**
* Problem: Counter II
* 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
*/

/**
* @param {integer} init
* @return { increment: Function, decrement: Function, reset: Function }
*/
var createCounter = function(init) {

};

/**
* const counter = createCounter(5)
* counter.increment(); // 6
* counter.reset(); // 5
* counter.decrement(); // 4
*/
```

TypeScript Solution:

```
/**
* Problem: Counter II
* 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
*/

type Counter = {
increment: () => number,
```

```
decrement: () => number,
reset: () => number,
}

function createCounter(init: number): Counter {

};

/**
* const counter = createCounter(5)
* counter.increment(); // 6
* counter.reset(); // 5
* counter.decrement(); // 4
*/
```