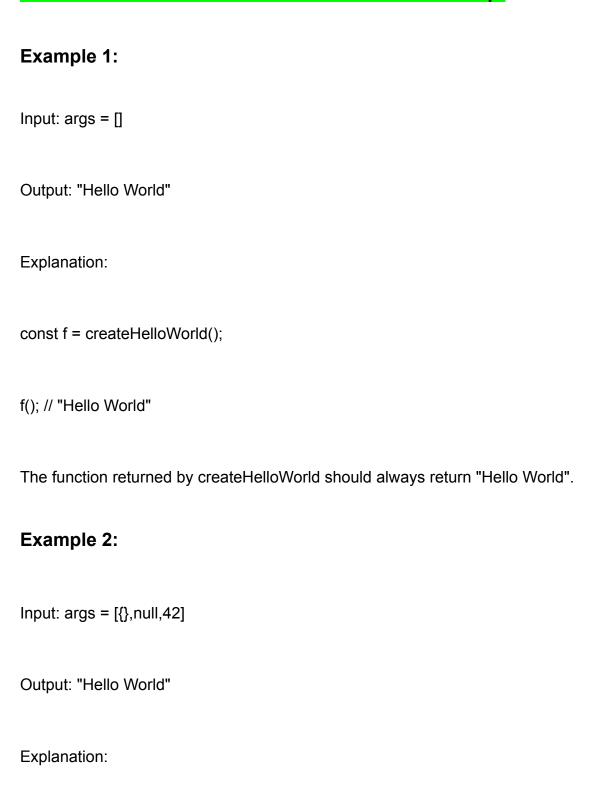
Solve the test Cases Given Below Based on Javascript



```
const f = createHelloWorld();
f({}, null, 42); // "Hello World"
```

Any arguments could be passed to the function but it should still always return

Constraints:

• 0 <= args.length <= 10

Given an integer n, return a counter function. This counter function initially returns n and then returns 1 more than the previous value every subsequent time it is called (n, n + 1, n + 2, etc)

Example:3

```
Input:
n = 10
["call","call","call"]
Output: [10,11,12]
Explanation:
counter() = 10 // The first time counter() is called, it returns n.
counter() = 11 // Returns 1 more than the previous time.
counter() = 12 // Returns 1 more than the previous time.
```

Example:4

```
Input:
n = -2
["call","call","call","call","call"]
Output: [-2,-1,0,1,2]
```

Explanation: counter() initially returns -2. Then increases after each sebsequent call.

Constraints:

- -1000 <= n <= 1000
- 0 <= calls.length <= 1000
- calls[i] === "call"

Write a function expect that helps developers test their code. It should take in any value val and return an object with the following two functions.

- toBe(val) accepts another value and returns true if the two values === each other. If they are not equal, it should throw an error "Not Equal".
- notToBe(val) accepts another value and returns true if the two values !== each other. If they are equal, it should throw an error "Equal".

Example:5

Input: func = () => expect(5).toBe(5)

Output: {"value": true}

Explanation: 5 === 5 so this expression returns true.

Example:6

Input: func = () => expect(5).toBe(null)

Output: {"error": "Not Equal"}

Explanation: 5 !== null so this expression throw the error "Not Equal".

Example:7

Input: func = () => expect(5).notToBe(null)

Output: {"value": true}

Explanation: 5 !== null so this expression returns true.

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:8

```
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:9

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