Problem 2621: Sleep

Problem Information

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

Problem Description

Given a positive integer

millis

, write an asynchronous function that sleeps for

millis

milliseconds. It can resolve any value.

Note

that

minor

deviation from

millis

in the actual sleep duration is acceptable.

Example 1:

Input:

millis = 100

Output:

100

Explanation:

It should return a promise that resolves after 100ms. let t = Date.now(); sleep(100).then(() => { console.log(Date.now() - t); // 100 });

Example 2:

Input:

millis = 200

Output:

200

Explanation:

It should return a promise that resolves after 200ms.

Constraints:

1 <= millis <= 1000

Code Snippets

JavaScript:

```
/**
* @param {number} millis
* @return {Promise}
*/
async function sleep(millis) {

}

/**
* let t = Date.now()
* sleep(100).then(() => console.log(Date.now() - t)) // 100
*/
```

TypeScript:

```
async function sleep(millis: number): Promise<void> {

}



/**
* let t = Date.now()
* sleep(100).then(() => console.log(Date.now() - t)) // 100
*/
```

Solutions

JavaScript Solution:

```
/**
* Problem: Sleep
* 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 {number} millis
* @return {Promise}
*/
async function sleep(millis) {

}
```

```
/**
* let t = Date.now()
* sleep(100).then(() => console.log(Date.now() - t)) // 100
*/
```

TypeScript Solution:

```
/**
* Problem: Sleep
* 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
*/

async function sleep(millis: number): Promise<void> {

}



/**
* let t = Date.now()
* sleep(100).then(() => console.log(Date.now() - t)) // 100
*/
```