Problem 2629: Function Composition

Problem Information

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

Problem Description

Given an array of functions

[f

1

, f

2

, f

3

, ..., f

n

]

, return a new function

fn

that is the

function composition

of the array of functions.

The

function composition

of

[f(x), g(x), h(x)]

is

fn(x) = f(g(h(x)))

.

The

function composition

of an empty list of functions is the

identity function

f(x) = x

.

You may assume each function in the array accepts one integer as input and returns one integer as output.

Example 1:

Input:

functions = [x => x + 1, x => x * x, x => 2 * x], x = 4

Output:

65

Explanation:

Evaluating from right to left ... Starting with x = 4. 2 * (4) = 8 (8) * (8) = 64 (64) + 1 = 65

Example 2:

Input:

functions = [x => 10 * x, x => 10 * x, x => 10 * x], x = 1

Output:

1000

Explanation:

Evaluating from right to left ... 10 * (1) = 10 10 * (10) = 100 10 * (100) = 1000

Example 3:

Input:

functions = [], x = 42

Output:

42

Explanation:

The composition of zero functions is the identity function

Constraints:

-1000 <= x <= 1000

0 <= functions.length <= 1000

all functions accept and return a single integer

Code Snippets

JavaScript:

```
/**
* @param {Function[]} functions
* @return {Function}
*/
var compose = function(functions) {

return function(x) {

}
};

/**
* const fn = compose([x => x + 1, x => 2 * x])
* fn(4) // 9
*/
```

TypeScript:

```
type F = (x: number) => number;

function compose(functions: F[]): F {

return function(x) {

}
};

/**
* const fn = compose([x => x + 1, x => 2 * x])
* fn(4) // 9
*/
```

Solutions

JavaScript Solution:

```
/**
* Problem: Function Composition
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {Function[]} functions
* @return {Function}
*/
var compose = function(functions) {

return function(x) {

}
};

/**
* const fn = compose([x => x + 1, x => 2 * x])
* fn(4) // 9
*/
```

TypeScript Solution:

```
/**
* Problem: Function Composition
* Difficulty: Easy
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
type F = (x: number) => number;

function compose(functions: F[]): F {

return function(x) {

}
};

/**
* const fn = compose([x => x + 1, x => 2 * x])
* fn(4) // 9
*/
```