Problem 2894: Divisible and Non-divisible Sums Difference

Problem Information

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

Problem Description

You are given positive integers

n

and

m

.

Define two integers as follows:

num1

: The sum of all integers in the range

[1, n]

(both

inclusive

) that are

not divisible

by

m

.

num2

: The sum of all integers in the range

[1, n]

(both

inclusive

) that are

divisible

by

m

.

Return

the integer

num1 - num2

.

Example 1:

Input:

n = 10, m = 3

Output:

19

Explanation:

In the given example: - Integers in the range [1, 10] that are not divisible by 3 are [1,2,4,5,7,8,10], num1 is the sum of those integers = 37. - Integers in the range [1, 10] that are divisible by 3 are [3,6,9], num2 is the sum of those integers = 18. We return 37 - 18 = 19 as the answer.

Example 2:

Input:

n = 5, m = 6

Output:

15

Explanation:

In the given example: - Integers in the range [1, 5] that are not divisible by 6 are [1,2,3,4,5], num1 is the sum of those integers = 15. - Integers in the range [1, 5] that are divisible by 6 are [], num2 is the sum of those integers = 0. We return 15 - 0 = 15 as the answer.

Example 3:

Input:

n = 5, m = 1

Output:

-15

Explanation:

In the given example: - Integers in the range [1, 5] that are not divisible by 1 are [], num1 is the sum of those integers = 0. - Integers in the range [1, 5] that are divisible by 1 are [1,2,3,4,5], num2 is the sum of those integers = 15. We return 0 - 15 = -15 as the answer.

Constraints:

1 <= n, m <= 1000

Code Snippets

C++:

```
class Solution {
public:
int differenceOfSums(int n, int m) {

}
};
```

Java:

```
class Solution {
public int differenceOfSums(int n, int m) {

}
}
```

Python3:

```
class Solution:
def differenceOfSums(self, n: int, m: int) -> int:
```

Python:

```
class Solution(object):
def differenceOfSums(self, n, m):
"""
:type n: int
:type m: int
```

```
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} m
* @return {number}
*/
var differenceOfSums = function(n, m) {

};
```

TypeScript:

```
function differenceOfSums(n: number, m: number): number {

};
```

C#:

```
public class Solution {
public int DifferenceOfSums(int n, int m) {

}
}
```

C:

```
int differenceOfSums(int n, int m) {

}
```

Go:

```
func differenceOfSums(n int, m int) int {

}
```

Kotlin:

```
class Solution {
fun differenceOfSums(n: Int, m: Int): Int {

}
}
```

Swift:

```
class Solution {
func differenceOfSums(_ n: Int, _ m: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn difference_of_sums(n: i32, m: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} m
# @return {Integer}
def difference_of_sums(n, m)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $m
* @return Integer
*/
function differenceOfSums($n, $m) {

}
```

```
}
```

Dart:

```
class Solution {
int differenceOfSums(int n, int m) {

}
}
```

Scala:

```
object Solution {
def differenceOfSums(n: Int, m: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec difference_of_sums(n :: integer, m :: integer) :: integer
def difference_of_sums(n, m) do

end
end
```

Erlang:

```
-spec difference_of_sums(N :: integer(), M :: integer()) -> integer().
difference_of_sums(N, M) ->
.
```

Racket:

```
(define/contract (difference-of-sums n m)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Divisible and Non-divisible Sums Difference
* Difficulty: Easy
* Tags: math
*
* 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
*/

class Solution {
public:
int differenceOfSums(int n, int m) {

}
};
```

Java Solution:

```
/**
* Problem: Divisible and Non-divisible Sums Difference
* Difficulty: Easy
* Tags: math
*
* 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
*/

class Solution {
public int differenceOfSums(int n, int m) {

}
}
```

Python3 Solution:

```
"""
Problem: Divisible and Non-divisible Sums Difference
Difficulty: Easy
Tags: math
```

```
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
"""

class Solution:
def differenceOfSums(self, n: int, m: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def differenceOfSums(self, n, m):
"""
:type n: int
:type m: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Divisible and Non-divisible Sums Difference
* Difficulty: Easy
* Tags: math
*
* 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} n
* @param {number} m
* @return {number}
*/
var differenceOfSums = function(n, m) {

};
```

TypeScript Solution:

```
/**
* Problem: Divisible and Non-divisible Sums Difference
* Difficulty: Easy
* Tags: math
*
* 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 differenceOfSums(n: number, m: number): number {

};
```

C# Solution:

```
/*
* Problem: Divisible and Non-divisible Sums Difference
* Difficulty: Easy
* Tags: math
*
* 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
*/

public class Solution {
public int DifferenceOfSums(int n, int m) {

}
}
```

C Solution:

```
/*
* Problem: Divisible and Non-divisible Sums Difference
* Difficulty: Easy
* Tags: math
*
* 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
*/

int differenceOfSums(int n, int m) {

}
```

Go Solution:

```
// Problem: Divisible and Non-divisible Sums Difference
// Difficulty: Easy
// Tags: math
//
// 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

func differenceOfSums(n int, m int) int {

}
```

Kotlin Solution:

```
class Solution {
fun differenceOfSums(n: Int, m: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func differenceOfSums(_ n: Int, _ m: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Divisible and Non-divisible Sums Difference
// Difficulty: Easy
// Tags: math
//
// 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

impl Solution {
pub fn difference_of_sums(n: i32, m: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} m
# @return {Integer}
def difference_of_sums(n, m)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $m
* @return Integer
*/
function differenceOfSums($n, $m) {

}
}
```

Dart Solution:

```
class Solution {
int differenceOfSums(int n, int m) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def differenceOfSums(n: Int, m: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec difference_of_sums(n :: integer, m :: integer) :: integer
def difference_of_sums(n, m) do

end
end
```

Erlang Solution:

```
-spec difference_of_sums(N :: integer(), M :: integer()) -> integer().
difference_of_sums(N, M) ->
.
```

Racket Solution:

```
(define/contract (difference-of-sums n m)
(-> exact-integer? exact-integer? exact-integer?)
)
```