Problem 3658: GCD of Odd and Even Sums

Problem Information

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

Problem Description

You are given an integer

n

. Your task is to compute the

GCD

(greatest common divisor) of two values:

sumOdd

: the sum of the first

n

odd numbers.

sumEven

: the sum of the first

n

even numbers.

Return the GCD of

sumOdd

and

sumEven

.

Example 1:

Input:

n = 4

Output:

4

Explanation:

Sum of the first 4 odd numbers

sumOdd = 1 + 3 + 5 + 7 = 16

Sum of the first 4 even numbers

sumEven = 2 + 4 + 6 + 8 = 20

Hence,

GCD(sumOdd, sumEven) = GCD(16, 20) = 4

.

Example 2:

Input:

n = 5

Output:

5

Explanation:

Sum of the first 5 odd numbers

sumOdd = 1 + 3 + 5 + 7 + 9 = 25

Sum of the first 5 even numbers

sumEven = 2 + 4 + 6 + 8 + 10 = 30

Hence,

GCD(sumOdd, sumEven) = GCD(25, 30) = 5

.

Constraints:

1 <= n <= 10■■■■■■■00

Code Snippets

C++:

```
class Solution {
public:
int gcdOfOddEvenSums(int n) {

}
};
```

Java:

```
class Solution {
public int gcdOfOddEvenSums(int n) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number} n
* @return {number}
*/
var gcdOfOddEvenSums = function(n) {

};
```

TypeScript:

```
function gcdOfOddEvenSums(n: number): number {

};
```

C#:

```
public class Solution {
public int GcdOfOddEvenSums(int n) {

}
}
```

C:

```
int gcdOfOddEvenSums(int n) {

}
```

Go:

```
func gcdOfOddEvenSums(n int) int {

}
```

Kotlin:

```
class Solution {
fun gcdOfOddEvenSums(n: Int): Int {

}
}
```

Swift:

```
class Solution {
func gcdOfOddEvenSums(_ n: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn gcd_of_odd_even_sums(n: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer}
def gcd_of_odd_even_sums(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function gcdOfOddEvenSums($n) {

}
}
```

Dart:

```
class Solution {
int gcdOfOddEvenSums(int n) {

}
}
```

Scala:

```
object Solution {
def gcdOfOddEvenSums(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec gcd_of_odd_even_sums(n :: integer) :: integer
def gcd_of_odd_even_sums(n) do

end
end
```

Erlang:

```
-spec gcd_of_odd_even_sums(N :: integer()) -> integer().
gcd_of_odd_even_sums(N) ->
.
```

Racket:

```
(define/contract (gcd-of-odd-even-sums n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: GCD of Odd and Even Sums
* 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 gcdOfOddEvenSums(int n) {

}
};
```

Java Solution:

```
/**
* Problem: GCD of Odd and Even Sums
* 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 gcdOfOddEvenSums(int n) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: GCD of Odd and Even Sums
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 gcdOfOddEvenSums(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: GCD of Odd and Even Sums
* 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
* @return {number}
*/
var gcdOfOddEvenSums = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: GCD of Odd and Even Sums
* 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 gcdOfOddEvenSums(n: number): number {

};
```

C# Solution:

```
/*
* Problem: GCD of Odd and Even Sums
* 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 GcdOfOddEvenSums(int n) {

}
```

```
}
```

C Solution:

```
/*
* Problem: GCD of Odd and Even Sums
* 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 gcdOfOddEvenSums(int n) {

}
```

Go Solution:

```
// Problem: GCD of Odd and Even Sums
// 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 gcdOfOddEvenSums(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun gcdOfOddEvenSums(n: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func gcdOfOddEvenSums(_ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: GCD of Odd and Even Sums
// 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 gcd_of_odd_even_sums(n: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def gcd_of_odd_even_sums(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function gcdOfOddEvenSums($n) {

}
}
```

Dart Solution:

```
class Solution {
int gcdOfOddEvenSums(int n) {

}
}
```

Scala Solution:

```
object Solution {
def gcdOfOddEvenSums(n: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec gcd_of_odd_even_sums(n :: integer) :: integer
def gcd_of_odd_even_sums(n) do

end
end
```

Erlang Solution:

```
-spec gcd_of_odd_even_sums(N :: integer()) -> integer().
gcd_of_odd_even_sums(N) ->
.
```

Racket Solution:

```
(define/contract (gcd-of-odd-even-sums n)
(-> exact-integer? exact-integer?)
)
```