Problem 371: Sum of Two Integers

Problem Information

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

Problem Description

Given two integers

a

and

b

, return

the sum of the two integers without using the operators

+

and

-

.

Example 1:

Input:

a = 1, b = 2

Output:

3

Example 2:

Input:

a = 2, b = 3

Output:

5

Constraints:

-1000 <= a, b <= 1000

Code Snippets

C++:

```
class Solution {
public:
int getSum(int a, int b) {

}
};
```

Java:

```
class Solution {
public int getSum(int a, int b) {

}
}
```

Python3:

```
class Solution:
def getSum(self, a: int, b: int) -> int:
```

Python:

```
class Solution(object):
def getSum(self, a, b):
"""
:type a: int
:type b: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} a
* @param {number} b
* @return {number}
*/
var getSum = function(a, b) {

};
```

TypeScript:

```
function getSum(a: number, b: number): number {

};
```

C#:

```
public class Solution {
public int GetSum(int a, int b) {

}
}
```

C:

```
int getSum(int a, int b) {

}
```

Go:

```
func getSum(a int, b int) int {

}
```

Kotlin:

```
class Solution {
fun getSum(a: Int, b: Int): Int {

}
}
```

Swift:

```
class Solution {
func getSum(_ a: Int, _ b: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn get_sum(a: i32, b: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} a
# @param {Integer} b
# @return {Integer}
def get_sum(a, b)

end
```

PHP:

```
class Solution {
```

```
/**
* @param Integer $a
* @param Integer $b
* @return Integer
*/
function getSum($a, $b) {

}
}
```

Dart:

```
class Solution {
int getSum(int a, int b) {

}
}
```

Scala:

```
object Solution {
def getSum(a: Int, b: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_sum(a :: integer, b :: integer) :: integer
def get_sum(a, b) do

end
end
```

Erlang:

```
-spec get_sum(A :: integer(), B :: integer()) -> integer().
get_sum(A, B) ->
.
```

Racket:

```
(define/contract (get-sum a b)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sum of Two Integers
* Difficulty: Medium
* 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 getSum(int a, int b) {

}
};
```

Java Solution:

```
/**
* Problem: Sum of Two Integers
* Difficulty: Medium
* 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 getSum(int a, int b) {

}
}
```

Python3 Solution:

```
"""
Problem: Sum of Two Integers
Difficulty: Medium
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 getSum(self, a: int, b: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getSum(self, a, b):
"""
:type a: int
:type b: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Sum of Two Integers
* Difficulty: Medium
* 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} a
* @param {number} b
* @return {number}
*/
var getSum = function(a, b) {

};
```

TypeScript Solution:

```
/**
* Problem: Sum of Two Integers
* Difficulty: Medium
* 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 getSum(a: number, b: number): number {

};
```

C# Solution:

```
/*
* Problem: Sum of Two Integers
* Difficulty: Medium
* 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 GetSum(int a, int b) {

}

}
```

C Solution:

```c
/*
* Problem: Sum of Two Integers
* Difficulty: Medium
* 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 getSum(int a, int b) {

}
```

Go Solution:

```go
// Problem: Sum of Two Integers
// Difficulty: Medium
// 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 getSum(a int, b int) int {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun getSum(a: Int, b: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func getSum(_ a: Int, _ b: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Sum of Two Integers
// Difficulty: Medium
// 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 get_sum(a: i32, b: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} a
# @param {Integer} b
# @return {Integer}
def get_sum(a, b)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $a
* @param Integer $b
* @return Integer
*/
function getSum($a, $b) {
}
}
```

Dart Solution:

```
class Solution {
int getSum(int a, int b) {

}
}
```

Scala Solution:

```
object Solution {
def getSum(a: Int, b: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_sum(a :: integer, b :: integer) :: integer
def get_sum(a, b) do

end
end
```

Erlang Solution:

```
-spec get_sum(A :: integer(), B :: integer()) -> integer().
get_sum(A, B) ->
.
```

Racket Solution:

```
(define/contract (get-sum a b)
(-> exact-integer? exact-integer? exact-integer?)
)
```