Problem 2443: Sum of Number and Its Reverse

Problem Information

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

Problem Description

Given a

non-negative

integer

num

, return

true

if

num

can be expressed as the sum of any

non-negative

integer and its reverse, or

false

otherwise.

Example 1:

Input:

num = 443

Output:

true

Explanation:

172 + 271 = 443 so we return true.

Example 2:

Input:

num = 63

Output:

false

Explanation:

63 cannot be expressed as the sum of a non-negative integer and its reverse so we return false.

Example 3:

Input:

num = 181

Output:

true

Explanation:

140 + 041 = 181 so we return true. Note that when a number is reversed, there may be leading zeros.

Constraints:

0 <= num <= 10

5

Code Snippets

C++:

```
class Solution {
public:
bool sumOfNumberAndReverse(int num) {

}
};
```

Java:

```
class Solution {
public boolean sumOfNumberAndReverse(int num) {

}
}
```

Python3:

```
class Solution:
def sumOfNumberAndReverse(self, num: int) -> bool:
```

Python:

```
class Solution(object):
def sumOfNumberAndReverse(self, num):
"""
```

```
:type num: int
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number} num
* @return {boolean}
*/
var sumOfNumberAndReverse = function(num) {

};
```

TypeScript:

```
function sumOfNumberAndReverse(num: number): boolean {

};
```

C#:

```
public class Solution {
public bool SumOfNumberAndReverse(int num) {

}
}
```

C:

```
bool sumOfNumberAndReverse(int num) {

}
```

Go:

```
func sumOfNumberAndReverse(num int) bool {

}
```

Kotlin:

```
class Solution {
fun sumOfNumberAndReverse(num: Int): Boolean {

}
}
```

Swift:

```
class Solution {
func sumOfNumberAndReverse(_ num: Int) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn sum_of_number_and_reverse(num: i32) -> bool {

}
}
```

Ruby:

```
# @param {Integer} num
# @return {Boolean}
def sum_of_number_and_reverse(num)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @return Boolean
*/
function sumOfNumberAndReverse($num) {

}
}
```

Dart:

```
class Solution {
bool sumOfNumberAndReverse(int num) {

}
}
```

Scala:

```
object Solution {
def sumOfNumberAndReverse(num: Int): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec sum_of_number_and_reverse(num :: integer) :: boolean
def sum_of_number_and_reverse(num) do

end
end
```

Erlang:

```
-spec sum_of_number_and_reverse(Num :: integer()) -> boolean().
sum_of_number_and_reverse(Num) ->
.
```

Racket:

```
(define/contract (sum-of-number-and-reverse num)
(-> exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sum of Number and Its Reverse
* 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:
bool sumOfNumberAndReverse(int num) {

}
};
```

Java Solution:

```
/**
* Problem: Sum of Number and Its Reverse
* 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 boolean sumOfNumberAndReverse(int num) {

}
}
```

Python3 Solution:

```
"""
Problem: Sum of Number and Its Reverse
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 sumOfNumberAndReverse(self, num: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def sumOfNumberAndReverse(self, num):
"""
:type num: int
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Sum of Number and Its Reverse
* 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} num
* @return {boolean}
*/
var sumOfNumberAndReverse = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Sum of Number and Its Reverse
* 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 sumOfNumberAndReverse(num: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Sum of Number and Its Reverse
* 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 bool SumOfNumberAndReverse(int num) {

}
}
```

C Solution:

```
/*
* Problem: Sum of Number and Its Reverse
* 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
```

```
*/

bool sumOfNumberAndReverse(int num) {

}
```

Go Solution:

```
// Problem: Sum of Number and Its Reverse
// 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 sumOfNumberAndReverse(num int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun sumOfNumberAndReverse(num: Int): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func sumOfNumberAndReverse(_ num: Int) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Sum of Number and Its Reverse
// 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 sum_of_number_and_reverse(num: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer} num
# @return {Boolean}
def sum_of_number_and_reverse(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @return Boolean
*/
function sumOfNumberAndReverse($num) {

}
}
```

Dart Solution:

```
class Solution {
bool sumOfNumberAndReverse(int num) {

}
}
```

Scala Solution:

```
object Solution {
def sumOfNumberAndReverse(num: Int): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec sum_of_number_and_reverse(num :: integer) :: boolean
def sum_of_number_and_reverse(num) do

end
end
```

Erlang Solution:

```
-spec sum_of_number_and_reverse(Num :: integer()) -> boolean().
sum_of_number_and_reverse(Num) ->
.
```

Racket Solution:

```
(define/contract (sum-of-number-and-reverse num)
(-> exact-integer? boolean?)
)
```