Problem 3622: Check Divisibility by Digit Sum and Product

Problem Information

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

Problem Description

You are given a positive integer

n

. Determine whether

n

is divisible by the

sum

of the following two values:

The

digit sum

of

n

(the sum of its digits).

The

digit

product

of

n

(the product of its digits).

Return

true

if

n

is divisible by this sum; otherwise, return

false

.

Example 1:

Input:

n = 99

Output:

true

Explanation:

Since 99 is divisible by the sum (9 + 9 = 18) plus product (9 * 9 = 81) of its digits (total 99), the output is true.

Example 2:

Input:

n = 23

Output:

false

Explanation:

Since 23 is not divisible by the sum (2 + 3 = 5) plus product (2 * 3 = 6) of its digits (total 11), the output is false.

Constraints:

1 <= n <= 10

6

Code Snippets

C++:

```
class Solution {
public:
bool checkDivisibility(int n) {

}
};
```

Java:

```
class Solution {
public boolean checkDivisibility(int n) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function checkDivisibility(n: number): boolean {

};
```

C#:

```
public class Solution {
public bool CheckDivisibility(int n) {

}
}
```

C:

```
bool checkDivisibility(int n) {

}
```

Go:

```
func checkDivisibility(n int) bool {

}
```

Kotlin:

```
class Solution {
fun checkDivisibility(n: Int): Boolean {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
```

```
* @param Integer $n
* @return Boolean
*/
function checkDivisibility($n) {

}
}
```

Dart:

```
class Solution {
bool checkDivisibility(int n) {

}
}
```

Scala:

```
object Solution {
def checkDivisibility(n: Int): Boolean = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec check_divisibility(N :: integer()) -> boolean().
check_divisibility(N) ->
.
```

Racket:

```
(define/contract (check-divisibility n)
(-> exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check Divisibility by Digit Sum and Product
* 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:
bool checkDivisibility(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Check Divisibility by Digit Sum and Product
* 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 boolean checkDivisibility(int n) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Check Divisibility by Digit Sum and Product
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 checkDivisibility(self, n: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check Divisibility by Digit Sum and Product
* 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 {boolean}
*/
var checkDivisibility = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Check Divisibility by Digit Sum and Product
* 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 checkDivisibility(n: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Check Divisibility by Digit Sum and Product
* 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 bool CheckDivisibility(int n) {

}
}
```

C Solution:

```
/*
* Problem: Check Divisibility by Digit Sum and Product
* 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
*/

bool checkDivisibility(int n) {

}
```

Go Solution:

```
// Problem: Check Divisibility by Digit Sum and Product
// 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 checkDivisibility(n int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun checkDivisibility(n: Int): Boolean {

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Check Divisibility by Digit Sum and Product
// 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 check_divisibility(n: i32) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
bool checkDivisibility(int n) {

}
}
```

Scala Solution:

```
object Solution {
def checkDivisibility(n: Int): Boolean = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec check_divisibility(N :: integer()) -> boolean().
check_divisibility(N) ->
.
```

Racket Solution:

```
(define/contract (check-divisibility n)
(-> exact-integer? boolean?)
)
```