Problem 1952: Three Divisors

Problem Information

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

Problem Description

Given an integer

n

, return

true

if

n

has

exactly three positive divisors

. Otherwise, return

false

.

An integer

m

is a

divisor

of

n

if there exists an integer

k

such that

n = k * m

.

Example 1:

Input:

n = 2

Output:

false

Explantion:

2 has only two divisors: 1 and 2.

Example 2:

Input:

n = 4

Output:

true

Explantion:

4 has three divisors: 1, 2, and 4.

Constraints:

1 <= n <= 10

4

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def isThree(self, n):
```

```
"""
:type n: int
:rtype: bool
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool isThree(int n) {

}
```

Go:

```
func isThree(n int) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (is-three n)
(-> exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Three Divisors
* 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 isThree(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Three Divisors
* 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 isThree(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Three Divisors
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 isThree(self, n: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Three Divisors
* 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 isThree = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Three Divisors
* 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 isThree(n: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Three Divisors
* 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 IsThree(int n) {

}
}
```

C Solution:

```
/*
* Problem: Three Divisors
* 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 isThree(int n) {

}
```

Go Solution:

```
// Problem: Three Divisors
// 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 isThree(n int) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Three Divisors
// 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 is_three(n: i32) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (is-three n)
(-> exact-integer? boolean?)
)
```