Problem 2729: Check if The Number is Fascinating

Problem Information

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

Problem Description

You are given an integer

n

that consists of exactly

3

digits.

We call the number

n

fascinating

if, after the following modification, the resulting number contains all the digits from

1

to

9

exactly

once and does not contain any

0

's:

Concatenate

n

with the numbers

2 * n

and

3 * n

.

Return

true

if

n

is fascinating, or

false

otherwise

.

Concatenating

two numbers means joining them together. For example, the concatenation of

121

and

371

is

121371

.

Example 1:

Input:

n = 192

Output:

true

Explanation:

We concatenate the numbers n = 192 and 2 * n = 384 and 3 * n = 576. The resulting number is 192384576. This number contains all the digits from 1 to 9 exactly once.

Example 2:

Input:

n = 100

Output:

false

Explanation:

We concatenate the numbers n = 100 and 2 * n = 200 and 3 * n = 300. The resulting number is 100200300. This number does not satisfy any of the conditions.

Constraints:

100 <= n <= 999

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool isFascinating(int n) {

}
```

Go:

```
func isFascinating(n int) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

```erlang
-spec is_fascinating(N :: integer()) -> boolean().
is_fascinating(N) ->
.
```

Racket:

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

Solutions

C++ Solution:

```cpp
/*
* Problem: Check if The Number is Fascinating
* Difficulty: Easy
* Tags: math, hash
*
```

```
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
bool isFascinating(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Check if The Number is Fascinating
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public boolean isFascinating(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Check if The Number is Fascinating
Difficulty: Easy
Tags: math, hash

Approach: Use hash map for O(1) lookups
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(n) for hash map
"""
```

```
class Solution:
def isFascinating(self, n: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Check if The Number is Fascinating
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Check if The Number is Fascinating
* Difficulty: Easy
* Tags: math, hash
```

```
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

function isFascinating(n: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if The Number is Fascinating
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

public class Solution {
public bool IsFascinating(int n) {

}
}
```

C Solution:

```
/*
* Problem: Check if The Number is Fascinating
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

bool isFascinating(int n) {
}
```

Go Solution:

```
// Problem: Check if The Number is Fascinating
// Difficulty: Easy
// Tags: math, hash
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map

func isFascinating(n int) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Check if The Number is Fascinating
// Difficulty: Easy
// Tags: math, hash
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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