

Problem 1056: Confusing Number

Problem Information

Difficulty: **Easy**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

A

confusing number

is a number that when rotated

180

degrees becomes a different number with

each digit valid

We can rotate digits of a number by

180

degrees to form new digits.

When

0

,

1

,

6

,

8

, and

9

are rotated

180

degrees, they become

0

,

1

,

9

,

8

, and

6

respectively.

When

2

,

3

,

4

,

5

, and

7

are rotated

180

degrees, they become

invalid

.

Note that after rotating a number, we can ignore leading zeros.

For example, after rotating

8000

, we have

0008

which is considered as just

8

Given an integer

n

, return

true

if it is a

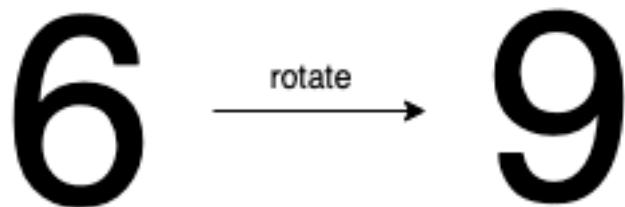
confusing number

, or

false

otherwise

Example 1:



Input:

$n = 6$

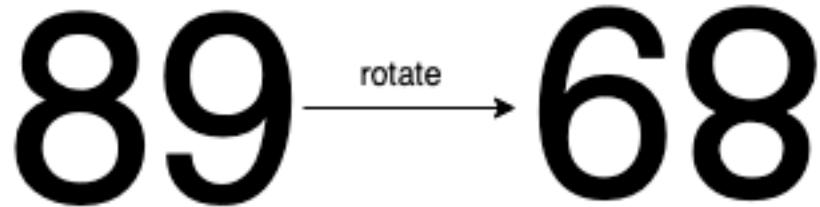
Output:

true

Explanation:

We get 9 after rotating 6, 9 is a valid number, and $9 \neq 6$.

Example 2:



Input:

$n = 89$

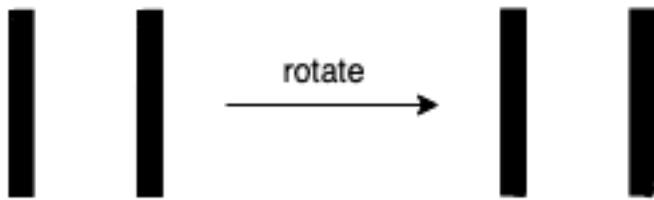
Output:

true

Explanation:

We get 68 after rotating 89, 68 is a valid number and $68 \neq 89$.

Example 3:



Input:

$n = 11$

Output:

false

Explanation:

We get 11 after rotating 11, 11 is a valid number but the value remains the same, thus 11 is not a confusing number

Constraints:

$0 \leq n \leq 10$

9

Code Snippets

C++:

```
class Solution {
public:
    bool confusingNumber(int n) {
        }
};
```

Java:

```
class Solution {  
    public boolean confusingNumber(int n) {  
  
    }  
}
```

Python3:

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

Python:

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

JavaScript:

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

TypeScript:

```
function confusingNumber(n: number): boolean {  
  
};
```

C#:

```
public class Solution {  
    public bool ConfusingNumber(int n) {  
  
    }  
}
```

C:

```
bool confusingNumber(int n) {  
}  
}
```

Go:

```
func confusingNumber(n int) bool {  
}  
}
```

Kotlin:

```
class Solution {  
    fun confusingNumber(n: Int): Boolean {  
        }  
        }  
}
```

Swift:

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

Rust:

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

Ruby:

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

PHP:

```
class Solution {  
  
    /**  
     * @param Integer $n  
     * @return Boolean  
     */  
    function confusingNumber($n) {  
  
    }  
}
```

Dart:

```
class Solution {  
  bool confusingNumber(int n) {  
  
  }  
}
```

Scala:

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

Elixir:

```
defmodule Solution do  
  @spec confusing_number(non_neg_integer) :: boolean  
  def confusing_number(n) do  
  
  end  
end
```

Erlang:

```
-spec confusing_number(non_neg_integer()) -> boolean().  
confusing_number(N) ->  
.
```

Racket:

```
(define/contract (confusing-number n)
  (-> exact-integer? boolean?))
```

Solutions

C++ Solution:

```
/*
 * Problem: Confusing Number
 * 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 confusingNumber(int n) {
        }
};
```

Java Solution:

```
/**
 * Problem: Confusing Number
 * 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 confusingNumber(int n) {
```

```
}
```

```
}
```

Python3 Solution:

```
"""
Problem: Confusing Number
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 confusingNumber(self, n: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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


```

JavaScript Solution:

```
/**
 * Problem: Confusing Number
 * 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 confusingNumber = function(n) {  
  
};
```

TypeScript Solution:

```
/**  
 * Problem: Confusing Number  
 * 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 confusingNumber(n: number): boolean {  
  
};
```

C# Solution:

```
/*  
 * Problem: Confusing Number  
 * 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 ConfusingNumber(int n) {  
  
    }
```

```
}
```

C Solution:

```
/*
 * Problem: Confusing Number
 * 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 confusingNumber(int n) {

}
```

Go Solution:

```
// Problem: Confusing Number
// 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 confusingNumber(n int) bool {

}
```

Kotlin Solution:

```
class Solution {
    fun confusingNumber(n: Int): Boolean {
        }
    }
```

Swift Solution:

```
class Solution {  
    func confusingNumber(_ n: Int) -> Bool {  
        //  
        //  
    }  
}
```

Rust Solution:

```
// Problem: Confusing Number  
// 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 confusing_number(n: i32) -> bool {  
        //  
        //  
    }  
}
```

Ruby Solution:

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

PHP Solution:

```
class Solution {  
  
    /**  
     * @param Integer $n  
     * @return Boolean  
     */  
    function confusingNumber($n) {  
  
    }  
}
```

Dart Solution:

```
class Solution {  
bool confusingNumber(int n) {  
  
}  
}
```

Scala Solution:

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

Elixir Solution:

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

Erlang Solution:

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

Racket Solution:

```
(define/contract (confusing-number n)  
(-> exact-integer? boolean?)  
)
```