Problem 3272: Find the Count of Good Integers

Problem Information

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

Problem Description

You are given two

positive

integers

n

and

k

.

An integer

x

is called

k-palindromic

if:

x

is a

palindrome

.

x

is divisible by

k

.

An integer is called

good

if its digits can be

rearranged

to form a

k-palindromic

integer. For example, for

k = 2

, 2020 can be rearranged to form the

k-palindromic

integer 2002, whereas 1010 cannot be rearranged to form a

k-palindromic

integer.

Return the count of

good

integers containing

n

digits.

Note

that

any

integer must

not

have leading zeros,

neither

before

nor

after rearrangement. For example, 1010

cannot

be rearranged to form 101.

Example 1:

Input:

n = 3, k = 5

Output:

27

Explanation:

Some

of the good integers are:

551 because it can be rearranged to form 515.

525 because it is already k-palindromic.

Example 2:

Input:

n = 1, k = 4

Output:

2

Explanation:

The two good integers are 4 and 8.

Example 3:

Input:

n = 5, k = 6

Output:

2468

Constraints:

1 <= n <= 10

1 <= k <= 9

Code Snippets

C++:

```
class Solution {
public:
long long countGoodIntegers(int n, int k) {

}
};
```

Java:

```
class Solution {
public long countGoodIntegers(int n, int k) {

}
}
```

Python3:

```
class Solution:
def countGoodIntegers(self, n: int, k: int) -> int:
```

Python:

```
class Solution(object):
def countGoodIntegers(self, n, k):
"""
:type n: int
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} k
* @return {number}
*/
var countGoodIntegers = function(n, k) {

};
```

TypeScript:

```
function countGoodIntegers(n: number, k: number): number {

};
```

C#:

```
public class Solution {
public long CountGoodIntegers(int n, int k) {

}
}
```

C:

```
long long countGoodIntegers(int n, int k) {

}
```

Go:

```
func countGoodIntegers(n int, k int) int64 {

}
```

Kotlin:

```
class Solution {
fun countGoodIntegers(n: Int, k: Int): Long {

}
```

```
}
```

Swift:

```
class Solution {
func countGoodIntegers(_ n: Int, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_good_integers(n: i32, k: i32) -> i64 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} k
# @return {Integer}
def count_good_integers(n, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return Integer
*/
function countGoodIntegers($n, $k) {

}
}
```

Dart:

```
class Solution {
int countGoodIntegers(int n, int k) {

}
}
```

Scala:

```
object Solution {
def countGoodIntegers(n: Int, k: Int): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_good_integers(n :: integer, k :: integer) :: integer
def count_good_integers(n, k) do

end
end
```

Erlang:

```
-spec count_good_integers(N :: integer(), K :: integer()) -> integer().
count_good_integers(N, K) ->
.
```

Racket:

```
(define/contract (count-good-integers n k)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Count of Good Integers
```

```
* Difficulty: Hard
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
long long countGoodIntegers(int n, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Count of Good Integers
* Difficulty: Hard
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public long countGoodIntegers(int n, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Count of Good Integers
Difficulty: Hard
Tags: string, math, hash

Approach: String manipulation with hash map or two pointers
```

```
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

```
class Solution(object):
def countGoodIntegers(self, n, k):
"""
:type n: int
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Count of Good Integers
* Difficulty: Hard
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number} n
* @param {number} k
* @return {number}
*/
var countGoodIntegers = function(n, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Count of Good Integers
* Difficulty: Hard
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function countGoodIntegers(n: number, k: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Count of Good Integers
* Difficulty: Hard
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public long CountGoodIntegers(int n, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find the Count of Good Integers
* Difficulty: Hard
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

long long countGoodIntegers(int n, int k) {

}
```

Go Solution:

```
// Problem: Find the Count of Good Integers
// Difficulty: Hard
// Tags: string, math, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func countGoodIntegers(n int, k int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun countGoodIntegers(n: Int, k: Int): Long {

}
}
```

Swift Solution:

```
class Solution {
func countGoodIntegers(_ n: Int, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Count of Good Integers
// Difficulty: Hard
// Tags: string, math, hash
```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn count_good_integers(n: i32, k: i32) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} k
# @return {Integer}
def count_good_integers(n, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return Integer
*/
function countGoodIntegers($n, $k) {

}
}
```

Dart Solution:

```
class Solution {
int countGoodIntegers(int n, int k) {

}
}
```

Scala Solution:

```
object Solution {
def countGoodIntegers(n: Int, k: Int): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_good_integers(n :: integer, k :: integer) :: integer
def count_good_integers(n, k) do

end
end
```

Erlang Solution:

```
-spec count_good_integers(N :: integer(), K :: integer()) -> integer().
count_good_integers(N, K) ->
.
```

Racket Solution:

```
(define/contract (count-good-integers n k)
(-> exact-integer? exact-integer? exact-integer?)
)
```