Problem 3581: Count Odd Letters from Number

Problem Information

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

Problem Description

You are given an integer

n

perform the following steps:

Convert each digit of

n

into its

lowercase English word

(e.g., 4 → "four", 1 → "one").

Concatenate

those words in the

original digit order

to form a string

s

.

Return the number of

distinct

characters in

s

that appear an

odd

number of times.

Example 1:

Input:

n = 41

Output:

5

Explanation:

41 $\rightarrow$

"fourone"

Characters with odd frequencies:

'f'

,

'u'

,

'r'

,

'n'

,

'e'

. Thus, the answer is 5.

Example 2:

Input:

n = 20

Output:

5

Explanation:

20 →

"twozero"

Characters with odd frequencies:

't'

,

'w'

,

'z'

,

'e'

,

'r'

. Thus, the answer is 5.

Constraints:

1 <= n <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int countOddLetters(int n) {

}
};
```

Java:

```
class Solution {
public int countOddLetters(int n) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function countOddLetters(n: number): number {

};
```

C#:

```
public class Solution {
public int CountOddLetters(int n) {

}
}
```

C:

```
int countOddLetters(int n) {

}
```

Go:

```
func countOddLetters(n int) int {

}
```

Kotlin:

```
class Solution {
fun countOddLetters(n: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
```

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

}
}
```

Dart:

```
class Solution {
int countOddLetters(int n) {

}
}
```

Scala:

```
object Solution {
def countOddLetters(n: Int): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec count_odd_letters(N :: integer()) -> integer().
count_odd_letters(N) ->
.
```

Racket:

```
(define/contract (count-odd-letters n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Odd Letters from Number
* Difficulty: Easy
* Tags: string, 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:
int countOddLetters(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Count Odd Letters from Number
* Difficulty: Easy
* Tags: string, 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 int countOddLetters(int n) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Count Odd Letters from Number
Difficulty: Easy
Tags: string, 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 countOddLetters(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Count Odd Letters from Number
* Difficulty: Easy
* Tags: string, 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
* @return {number}
*/
var countOddLetters = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Odd Letters from Number
* Difficulty: Easy
* Tags: string, 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 countOddLetters(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Count Odd Letters from Number
* Difficulty: Easy
* Tags: string, 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 int CountOddLetters(int n) {

}
}
```

C Solution:

```
/*
* Problem: Count Odd Letters from Number
* Difficulty: Easy
* Tags: string, 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
*/

int countOddLetters(int n) {

}
```

Go Solution:

```
// Problem: Count Odd Letters from Number
// Difficulty: Easy
// Tags: string, 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 countOddLetters(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countOddLetters(n: Int): Int {

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Count Odd Letters from Number
// Difficulty: Easy
// Tags: string, 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_odd_letters(n: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int countOddLetters(int n) {

}
}
```

Scala Solution:

```
object Solution {
def countOddLetters(n: Int): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec count_odd_letters(N :: integer()) -> integer().
count_odd_letters(N) ->
.
```

Racket Solution:

```
(define/contract (count-odd-letters n)
(-> exact-integer? exact-integer?)
)
```