Problem 1342: Number of Steps to Reduce a Number to Zero

Problem Information

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

Problem Description

Given an integer

num

, return

the number of steps to reduce it to zero

.

In one step, if the current number is even, you have to divide it by

2

, otherwise, you have to subtract

1

from it.

Example 1:

Input:

num = 14

Output:

6

Explanation:

Step 1) 14 is even; divide by 2 and obtain 7. Step 2) 7 is odd; subtract 1 and obtain 6. Step 3) 6 is even; divide by 2 and obtain 3. Step 4) 3 is odd; subtract 1 and obtain 2. Step 5) 2 is even; divide by 2 and obtain 1. Step 6) 1 is odd; subtract 1 and obtain 0.

Example 2:

Input:

num = 8

Output:

4

Explanation:

Step 1) 8 is even; divide by 2 and obtain 4. Step 2) 4 is even; divide by 2 and obtain 2. Step 3) 2 is even; divide by 2 and obtain 1. Step 4) 1 is odd; subtract 1 and obtain 0.

Example 3:

Input:

num = 123

Output:

12

Constraints:

0 <= num <= 10

Code Snippets

C++:

```
class Solution {
public:
int numberOfSteps(int num) {

}
};
```

Java:

```
class Solution {
public int numberOfSteps(int num) {

}
}
```

Python3:

```
class Solution:
def numberOfSteps(self, num: int) -> int:
```

Python:

```
class Solution(object):
def numberOfSteps(self, num):
"""
:type num: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} num
* @return {number}
*/
```

```
var numberOfSteps = function(num) {

};
```

TypeScript:

```
function numberOfSteps(num: number): number {

};
```

C#:

```
public class Solution {
public int NumberOfSteps(int num) {

}
}
```

C:

```
int numberOfSteps(int num) {

}
```

Go:

```
func numberOfSteps(num int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfSteps(num: Int): Int {

}
}
```

Swift:

```
class Solution {
func numberOfSteps(_ num: Int) -> Int {
```

```
}
}
```

Rust:

```
impl Solution {
pub fn number_of_steps(num: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} num
# @return {Integer}
def number_of_steps(num)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function numberOfSteps($num) {

}
}
```

Dart:

```
class Solution {
int numberOfSteps(int num) {

}
}
```

Scala:

```
object Solution {
def numberOfSteps(num: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_steps(num :: integer) :: integer
def number_of_steps(num) do

end
end
```

Erlang:

```
-spec number_of_steps(Num :: integer()) -> integer().
number_of_steps(Num) ->
.
```

Racket:

```
(define/contract (number-of-steps num)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Steps to Reduce a Number to Zero
* 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:
int numberOfSteps(int num) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Steps to Reduce a Number to Zero
* 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 int numberOfSteps(int num) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Steps to Reduce a Number to Zero
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 numberOfSteps(self, num: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfSteps(self, num):
"""
:type num: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Steps to Reduce a Number to Zero
* 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} num
* @return {number}
*/
var numberOfSteps = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Steps to Reduce a Number to Zero
* 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 numberOfSteps(num: number): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Number of Steps to Reduce a Number to Zero
* 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 int NumberOfSteps(int num) {

}
}
```

C Solution:

```
/*
* Problem: Number of Steps to Reduce a Number to Zero
* 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
*/

int numberOfSteps(int num) {

}
```

Go Solution:

```
// Problem: Number of Steps to Reduce a Number to Zero
// 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 numberOfSteps(num int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfSteps(num: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfSteps(_ num: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Steps to Reduce a Number to Zero
// 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 number_of_steps(num: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} num
# @return {Integer}
def number_of_steps(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function numberOfSteps($num) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfSteps(int num) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfSteps(num: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_steps(num :: integer) :: integer
def number_of_steps(num) do
```

```
end
end
```

Erlang Solution:

```
-spec number_of_steps(Num :: integer()) -> integer().
number_of_steps(Num) ->
.
```

Racket Solution:

```
(define/contract (number-of-steps num)
(-> exact-integer? exact-integer?)
)
```