

# Problem 2180: Count Integers With Even Digit Sum

## Problem Information

Difficulty: Easy

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

Given a positive integer

`num`

, return

the number of positive integers

less than or equal to

`num`

whose digit sums are

even

.

The

digit sum

of a positive integer is the sum of all its digits.

Example 1:

Input:

num = 4

Output:

2

Explanation:

The only integers less than or equal to 4 whose digit sums are even are 2 and 4.

Example 2:

Input:

num = 30

Output:

14

Explanation:

The 14 integers less than or equal to 30 whose digit sums are even are 2, 4, 6, 8, 11, 13, 15, 17, 19, 20, 22, 24, 26, and 28.

Constraints:

$1 \leq \text{num} \leq 1000$

## Code Snippets

**C++:**

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

```
}  
};
```

### Java:

```
class Solution {  
    public int countEven(int num) {  
  
    }  
}
```

### Python3:

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

### Python:

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

### JavaScript:

```
/**  
 * @param {number} num  
 * @return {number}  
 */  
var countEven = function(num) {  
  
};
```

### TypeScript:

```
function countEven(num: number): number {  
  
};
```

**C#:**

```
public class Solution {  
    public int CountEven(int num) {  
  
    }  
}
```

**C:**

```
int countEven(int num) {  
  
}
```

**Go:**

```
func countEven(num int) int {  
  
}
```

**Kotlin:**

```
class Solution {  
    fun countEven(num: Int): Int {  
  
    }  
}
```

**Swift:**

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

**Rust:**

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

## Ruby:

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

end
```

## PHP:

```
class Solution {

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

    }

}
```

## Dart:

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

  }
}
```

## Scala:

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

  }
}
```

## Elixir:

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

```
end  
end
```

### Erlang:

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

### Racket:

```
(define/contract (count-even num)  
  (-> exact-integer? exact-integer?)  
)
```

## Solutions

### C++ Solution:

```
/*  
 * Problem: Count Integers With Even Digit Sum  
 * 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 countEven(int num) {  
  
    }  
};
```

### Java Solution:

```
/**  
 * Problem: Count Integers With Even Digit Sum
```

```

* 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 countEven(int num) {

}
}

```

### Python3 Solution:

```

"""
Problem: Count Integers With Even Digit Sum
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 countEven(self, num: int) -> int:
        # TODO: Implement optimized solution
        pass

```

### Python Solution:

```

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

```

### JavaScript Solution:

```

/**
 * Problem: Count Integers With Even Digit Sum
 * 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 countEven = function(num) {

};

```

### TypeScript Solution:

```

/**
 * Problem: Count Integers With Even Digit Sum
 * 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 countEven(num: number): number {

};

```

### C# Solution:

```

/*
 * Problem: Count Integers With Even Digit Sum
 * 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 CountEven(int num) {

}

}

```

### C Solution:

```

/*
* Problem: Count Integers With Even Digit Sum
* 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 countEven(int num) {

}

```

### Go Solution:

```

// Problem: Count Integers With Even Digit Sum
// 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 countEven(num int) int {

}

```

### Kotlin Solution:

```

class Solution {
    fun countEven(num: Int): Int {

    }
}

```

### Swift Solution:

```

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

    }
}

```

### Rust Solution:

```

// Problem: Count Integers With Even Digit Sum
// 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 count_even(num: i32) -> i32 {

    }
}

```

### Ruby Solution:

```

# @param {Integer} num
# @return {Integer}
def count_even(num)

end

```

### PHP Solution:

```

class Solution {

```

```

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

}
}

```

### Dart Solution:

```

class Solution {
  int countEven(int num) {

  }
}

```

### Scala Solution:

```

object Solution {
  def countEven(num: Int): Int = {

  }
}

```

### Elixir Solution:

```

defmodule Solution do
  @spec count_even(num :: integer) :: integer
  def count_even(num) do

  end
end

```

### Erlang Solution:

```

-spec count_even(Num :: integer()) -> integer().
count_even(Num) ->
.

```

### Racket Solution:

```
(define/contract (count-even num)
  (-> exact-integer? exact-integer?)
)
```