

Problem 2455: Average Value of Even Numbers That Are Divisible by Three

Problem Information

Difficulty: [Easy](#)

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given an integer array

nums

of

positive

integers, return

the average value of all even integers that are divisible by

3

.

Note that the

average

of

n

elements is the

sum

of the

n

elements divided by

n

and

rounded down

to the nearest integer.

Example 1:

Input:

nums = [1,3,6,10,12,15]

Output:

9

Explanation:

6 and 12 are even numbers that are divisible by 3. $(6 + 12) / 2 = 9$.

Example 2:

Input:

nums = [1,2,4,7,10]

Output:

0

Explanation:

There is no single number that satisfies the requirement, so return 0.

Constraints:

$1 \leq \text{nums.length} \leq 1000$

$1 \leq \text{nums}[i] \leq 1000$

Code Snippets

C++:

```
class Solution {  
public:  
    int averageValue(vector<int>& nums) {  
  
    }  
};
```

Java:

```
class Solution {  
    public int averageValue(int[] nums) {  
  
    }  
}
```

Python3:

```
class Solution:  
    def averageValue(self, nums: List[int]) -> int:
```

Python:

```
class Solution(object):  
    def averageValue(self, nums):
```

```
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
 * @param {number[]} nums
 * @return {number}
 */
var averageValue = function(nums) {

};
```

TypeScript:

```
function averageValue(nums: number[]): number {

};
```

C#:

```
public class Solution {
    public int AverageValue(int[] nums) {

    }
}
```

C:

```
int averageValue(int* nums, int numsSize) {

}
```

Go:

```
func averageValue(nums []int) int {

}
```

Kotlin:

```

class Solution {
    fun averageValue(nums: IntArray): Int {

    }
}

```

Swift:

```

class Solution {
    func averageValue(_ nums: [Int]) -> Int {

    }
}

```

Rust:

```

impl Solution {
    pub fn average_value(nums: Vec<i32>) -> i32 {

    }
}

```

Ruby:

```

# @param {Integer[]} nums
# @return {Integer}
def average_value(nums)

end

```

PHP:

```

class Solution {

    /**
     * @param Integer[] $nums
     * @return Integer
     */
    function averageValue($nums) {

    }
}

```

Dart:

```
class Solution {  
  int averageValue(List<int> nums) {  
  
  }  
}
```

Scala:

```
object Solution {  
  def averageValue(nums: Array[Int]): Int = {  
  
  }  
}
```

Elixir:

```
defmodule Solution do  
  @spec average_value(nums :: [integer]) :: integer  
  def average_value(nums) do  
  
  end  
end
```

Erlang:

```
-spec average_value(Nums :: [integer()]) -> integer().  
average_value(Nums) ->  
.
```

Racket:

```
(define/contract (average-value nums)  
  (-> (listof exact-integer?) exact-integer?)  
  )
```

Solutions

C++ Solution:

```

/*
 * Problem: Average Value of Even Numbers That Are Divisible by Three
 * Difficulty: Easy
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public:
    int averageValue(vector<int>& nums) {

    }
};

```

Java Solution:

```

/**
 * Problem: Average Value of Even Numbers That Are Divisible by Three
 * Difficulty: Easy
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
    public int averageValue(int[] nums) {

    }
}

```

Python3 Solution:

```

"""
Problem: Average Value of Even Numbers That Are Divisible by Three
Difficulty: Easy
Tags: array, math

```

```

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
    def averageValue(self, nums: List[int]) -> int:
        # TODO: Implement optimized solution
        pass

```

Python Solution:

```

class Solution(object):
    def averageValue(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """

```

JavaScript Solution:

```

/**
 * Problem: Average Value of Even Numbers That Are Divisible by Three
 * Difficulty: Easy
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * @param {number[]} nums
 * @return {number}
 */
var averageValue = function(nums) {

};

```

TypeScript Solution:


```

/**
 * Problem: Average Value of Even Numbers That Are Divisible by Three
 * Difficulty: Easy
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

function averageValue(nums: number[]): number {

};

```

C# Solution:

```

/*
 * Problem: Average Value of Even Numbers That Are Divisible by Three
 * Difficulty: Easy
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

public class Solution {
    public int AverageValue(int[] nums) {

    }
}

```

C Solution:

```

/*
 * Problem: Average Value of Even Numbers That Are Divisible by Three
 * Difficulty: Easy
 * Tags: array, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach

```

```
*/

int averageValue(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Average Value of Even Numbers That Are Divisible by Three
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func averageValue(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
    fun averageValue(nums: IntArray): Int {

    }
}
```

Swift Solution:

```
class Solution {
    func averageValue(_ nums: [Int]) -> Int {

    }
}
```

Rust Solution:

```
// Problem: Average Value of Even Numbers That Are Divisible by Three
// Difficulty: Easy
// Tags: array, math
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
    pub fn average_value(nums: Vec<i32>) -> i32 {

    }
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def average_value(nums)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer[] $nums
     * @return Integer
     */
    function averageValue($nums) {

    }
}
```

Dart Solution:

```
class Solution {
    int averageValue(List<int> nums) {

    }
}
```

Scala Solution:

```
object Solution {  
  def averageValue(nums: Array[Int]): Int = {  
  
  }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec average_value(nums :: [integer]) :: integer  
  def average_value(nums) do  
  
  end  
end
```

Erlang Solution:

```
-spec average_value(Nums :: [integer()]) -> integer().  
average_value(Nums) ->  
.
```

Racket Solution:

```
(define/contract (average-value nums)  
  (-> (listof exact-integer?) exact-integer?)  
)
```