Problem 2413: Smallest Even Multiple

Problem Information

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

Problem Description

Given a

positive

integer

n

, return

the smallest positive integer that is a multiple of

both

2

and

n

.

Example 1:

Input:

n = 5

Output:

10

Explanation:

The smallest multiple of both 5 and 2 is 10.

Example 2:

Input:

n = 6

Output:

6

Explanation:

The smallest multiple of both 6 and 2 is 6. Note that a number is a multiple of itself.

Constraints:

1 <= n <= 150

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public int SmallestEvenMultiple(int n) {
}
}
```

C:

```
int smallestEvenMultiple(int n) {

}
```

Go:

```
func smallestEvenMultiple(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (smallest-even-multiple n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Smallest Even Multiple
* 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 smallestEvenMultiple(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest Even Multiple
* 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 smallestEvenMultiple(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Even Multiple
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 smallestEvenMultiple(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Smallest Even Multiple
* 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} n
* @return {number}
*/
var smallestEvenMultiple = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Even Multiple
* 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 smallestEvenMultiple(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Smallest Even Multiple
* 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 SmallestEvenMultiple(int n) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Even Multiple
* 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 smallestEvenMultiple(int n) {

}
```

Go Solution:

```
// Problem: Smallest Even Multiple
// 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 smallestEvenMultiple(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Smallest Even Multiple
// 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 smallest_even_multiple(n: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (smallest-even-multiple n)
(-> exact-integer? exact-integer?)
)
```