Problem 3492: Maximum Containers on a Ship

Problem Information

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

Problem Description

You are given a positive integer

n

representing an

n x n

cargo deck on a ship. Each cell on the deck can hold one container with a weight of

exactly

w

.

However, the total weight of all containers, if loaded onto the deck, must not exceed the ship's maximum weight capacity,

maxWeight

.

Return the

maximum

number of containers that can be loaded onto the ship.

Example 1:

Input:

n = 2, w = 3, maxWeight = 15

Output:

4

Explanation:

The deck has 4 cells, and each container weighs 3. The total weight of loading all containers is 12, which does not exceed

maxWeight

.

Example 2:

Input:

n = 3, w = 5, maxWeight = 20

Output:

4

Explanation:

The deck has 9 cells, and each container weighs 5. The maximum number of containers that can be loaded without exceeding

maxWeight

is 4.

Constraints:

1 <= n <= 1000

1 <= w <= 1000

1 <= maxWeight <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int maxContainers(int n, int w, int maxWeight) {

}
};
```

Java:

```
class Solution {
public int maxContainers(int n, int w, int maxWeight) {

}
}
```

Python3:

```
class Solution:
def maxContainers(self, n: int, w: int, maxWeight: int) -> int:
```

Python:

```
class Solution(object):
def maxContainers(self, n, w, maxWeight):
"""
:type n: int
```

```
:type w: int
:type maxWeight: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} w
* @param {number} maxWeight
* @return {number}
*/
var maxContainers = function(n, w, maxWeight) {

};
```

TypeScript:

```
function maxContainers(n: number, w: number, maxWeight: number): number {

};
```

C#:

```
public class Solution {
public int MaxContainers(int n, int w, int maxWeight) {

}
}
```

C:

```
int maxContainers(int n, int w, int maxWeight) {

}
```

Go:

```
func maxContainers(n int, w int, maxWeight int) int {

}
```

Kotlin:

```
class Solution {
fun maxContainers(n: Int, w: Int, maxWeight: Int): Int {

}
}
```

Swift:

```
class Solution {
func maxContainers(_ n: Int, _ w: Int, _ maxWeight: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_containers(n: i32, w: i32, max_weight: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} w
# @param {Integer} max_weight
# @return {Integer}
def max_containers(n, w, max_weight)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $w
* @param Integer $maxWeight
* @return Integer
*/
function maxContainers($n, $w, $maxWeight) {

}
}
```

Dart:

```
class Solution {
int maxContainers(int n, int w, int maxWeight) {

}
}
```

Scala:

```
object Solution {
def maxContainers(n: Int, w: Int, maxWeight: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_containers(n :: integer, w :: integer, max_weight :: integer) :: integer
def max_containers(n, w, max_weight) do

end
end
```

Erlang:

```
-spec max_containers(N :: integer(), W :: integer(), MaxWeight :: integer()) -> integer().
max_containers(N, W, MaxWeight) ->
.
```

Racket:

```
(define/contract (max-containers n w maxWeight)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Containers on a Ship
* 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 maxContainers(int n, int w, int maxWeight) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Containers on a Ship
* 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 maxContainers(int n, int w, int maxWeight) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Containers on a Ship
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 maxContainers(self, n: int, w: int, maxWeight: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxContainers(self, n, w, maxWeight):
"""
:type n: int
:type w: int
:type maxWeight: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Containers on a Ship
* 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
* @param {number} w
* @param {number} maxWeight
* @return {number}
*/
var maxContainers = function(n, w, maxWeight) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Containers on a Ship
* 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 maxContainers(n: number, w: number, maxWeight: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Containers on a Ship
* 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 MaxContainers(int n, int w, int maxWeight) {
}
}
```

C Solution:

```
/*
* Problem: Maximum Containers on a Ship
* 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 maxContainers(int n, int w, int maxWeight) {

}
```

Go Solution:

```
// Problem: Maximum Containers on a Ship
// 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 maxContainers(n int, w int, maxWeight int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxContainers(n: Int, w: Int, maxWeight: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxContainers(_ n: Int, _ w: Int, _ maxWeight: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Containers on a Ship
// 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 max_containers(n: i32, w: i32, max_weight: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} w
# @param {Integer} max_weight
# @return {Integer}
def max_containers(n, w, max_weight)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $w
* @param Integer $maxWeight
* @return Integer
*/
function maxContainers($n, $w, $maxWeight) {

}
}
```

Dart Solution:

```
class Solution {
int maxContainers(int n, int w, int maxWeight) {

}
}
```

Scala Solution:

```
object Solution {
def maxContainers(n: Int, w: Int, maxWeight: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_containers(n :: integer, w :: integer, max_weight :: integer) ::
integer
def max_containers(n, w, max_weight) do

end
end
```

Erlang Solution:

```
-spec max_containers(N :: integer(), W :: integer(), MaxWeight :: integer())
-> integer().
max_containers(N, W, MaxWeight) ->
.
```

Racket Solution:

```
(define/contract (max-containers n w maxWeight)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```