# Problem 2525: Categorize Box According to Criteria

## Problem Information

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

## Problem Description

Given four integers

length

,

width

,

height

, and

mass

, representing the dimensions and mass of a box, respectively, return

a string representing the

category

of the box

.

The box is

"Bulky"

if:

Any

of the dimensions of the box is greater or equal to

10

4

.

Or, the

volume

of the box is greater or equal to

10

9

.

If the mass of the box is greater or equal to

100

, it is

"Heavy".

If the box is both

"Bulky"

and

"Heavy"

, then its category is

"Both"

.

If the box is neither

"Bulky"

nor

"Heavy"

, then its category is

"Neither"

.

If the box is

"Bulky"

but not

"Heavy"

, then its category is

"Bulky"

.

If the box is

"Heavy"

but not

"Bulky"

, then its category is

"Heavy"

.

Note

that the volume of the box is the product of its length, width and height.

Example 1:

Input:

length = 1000, width = 35, height = 700, mass = 300

Output:

"Heavy"

Explanation:

None of the dimensions of the box is greater or equal to 10

4

. Its volume = 24500000 <= 10

9

. So it cannot be categorized as "Bulky". However mass >= 100, so the box is "Heavy". Since the box is not "Bulky" but "Heavy", we return "Heavy".

Example 2:

Input:

length = 200, width = 50, height = 800, mass = 50

Output:

"Neither"

Explanation:

None of the dimensions of the box is greater or equal to 10

4

. Its volume = 8 * 10

6

<= 10

9

. So it cannot be categorized as "Bulky". Its mass is also less than 100, so it cannot be categorized as "Heavy" either. Since its neither of the two above categories, we return "Neither".

Constraints:

1 <= length, width, height <= 10

5

1 <= mass <= 10

## Code Snippets

**C++:**

```
class Solution {
public:
string categorizeBox(int length, int width, int height, int mass) {

}
};
```

**Java:**

```
class Solution {
public String categorizeBox(int length, int width, int height, int mass) {

}
}
```

**Python3:**

```
class Solution:
def categorizeBox(self, length: int, width: int, height: int, mass: int) ->
str:
```

**Python:**

```
class Solution(object):
def categorizeBox(self, length, width, height, mass):
"""
:type length: int
:type width: int
:type height: int
:type mass: int
:rtype: str
"""
```

**JavaScript:**

```
/**
* @param {number} length
* @param {number} width
* @param {number} height
* @param {number} mass
* @return {string}
*/
var categorizeBox = function(length, width, height, mass) {

};
```

**TypeScript:**

```
function categorizeBox(length: number, width: number, height: number, mass:
number): string {

};
```

**C#:**

```
public class Solution {
public string CategorizeBox(int length, int width, int height, int mass) {

}
}
```

**C:**

```
char* categorizeBox(int length, int width, int height, int mass) {

}
```

**Go:**

```
func categorizeBox(length int, width int, height int, mass int) string {

}
```

**Kotlin:**

```
class Solution {
fun categorizeBox(length: Int, width: Int, height: Int, mass: Int): String {
```

```
}
}
```

**Swift:**

```
class Solution {
func categorizeBox(_ length: Int, _ width: Int, _ height: Int, _ mass: Int)
-> String {

}
}
```

**Rust:**

```
impl Solution {
pub fn categorize_box(length: i32, width: i32, height: i32, mass: i32) ->
String {

}
}
```

**Ruby:**

```
# @param {Integer} length
# @param {Integer} width
# @param {Integer} height
# @param {Integer} mass
# @return {String}
def categorize_box(length, width, height, mass)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer $length
* @param Integer $width
* @param Integer $height
* @param Integer $mass
* @return String
```

```
*/
function categorizeBox($length, $width, $height, $mass) {

}
}
```

**Dart:**

```
class Solution {
String categorizeBox(int length, int width, int height, int mass) {

}
}
```

**Scala:**

```
object Solution {
def categorizeBox(length: Int, width: Int, height: Int, mass: Int): String =
{

}
}
```

**Elixir:**

```
defmodule Solution do
@spec categorize_box(length :: integer, width :: integer, height :: integer,
mass :: integer) :: String.t
def categorize_box(length, width, height, mass) do

end
end
```

**Erlang:**

```
-spec categorize_box(Length :: integer(), Width :: integer(), Height ::
integer(), Mass :: integer()) -> unicode:unicode_binary().
categorize_box(Length, Width, Height, Mass) ->
.
```

**Racket:**

```
(define/contract (categorize-box length width height mass)
(-> exact-integer? exact-integer? exact-integer? exact-integer? string?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Categorize Box According to Criteria
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string categorizeBox(int length, int width, int height, int mass) {

}
};
```

### Java Solution:

```
/**
* Problem: Categorize Box According to Criteria
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String categorizeBox(int length, int width, int height, int mass) {

}
```

```
}
```

### Python3 Solution:

```
"""
Problem: Categorize Box According to Criteria
Difficulty: Easy
Tags: string, math

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def categorizeBox(self, length: int, width: int, height: int, mass: int) ->
str:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
class Solution(object):
def categorizeBox(self, length, width, height, mass):
"""
:type length: int
:type width: int
:type height: int
:type mass: int
:rtype: str
"""
```

### JavaScript Solution:

```
/**
* Problem: Categorize Box According to Criteria
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} length
* @param {number} width
* @param {number} height
* @param {number} mass
* @return {string}
*/
var categorizeBox = function(length, width, height, mass) {

};
```

**TypeScript Solution:**

```
/**
* Problem: Categorize Box According to Criteria
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function categorizeBox(length: number, width: number, height: number, mass:
number): string {

};
```

**C# Solution:**

```
/*
* Problem: Categorize Box According to Criteria
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

public class Solution {
public string CategorizeBox(int length, int width, int height, int mass) {

}
}
```

**C Solution:**

```
/*
* Problem: Categorize Box According to Criteria
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* categorizeBox(int length, int width, int height, int mass) {

}
```

**Go Solution:**

```
// Problem: Categorize Box According to Criteria
// Difficulty: Easy
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func categorizeBox(length int, width int, height int, mass int) string {

}
```

**Kotlin Solution:**

```
class Solution {
fun categorizeBox(length: Int, width: Int, height: Int, mass: Int): String {

}
}
```

**Swift Solution:**

```
class Solution {
func categorizeBox(_ length: Int, _ width: Int, _ height: Int, _ mass: Int)
-> String {

}
}
```

**Rust Solution:**

```
// Problem: Categorize Box According to Criteria
// Difficulty: Easy
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn categorize_box(length: i32, width: i32, height: i32, mass: i32) ->
String {

}
}
```

**Ruby Solution:**

```
# @param {Integer} length
# @param {Integer} width
# @param {Integer} height
# @param {Integer} mass
# @return {String}
def categorize_box(length, width, height, mass)

end
```

**PHP Solution:**

```
class Solution {

/**
* @param Integer $length
* @param Integer $width
* @param Integer $height
* @param Integer $mass
* @return String
*/
function categorizeBox($length, $width, $height, $mass) {

}
}
```

**Dart Solution:**

```
class Solution {
String categorizeBox(int length, int width, int height, int mass) {

}
}
```

**Scala Solution:**

```
object Solution {
def categorizeBox(length: Int, width: Int, height: Int, mass: Int): String =
{

}
}
```

**Elixir Solution:**

```
defmodule Solution do
@spec categorize_box(length :: integer, width :: integer, height :: integer,
mass :: integer) :: String.t
def categorize_box(length, width, height, mass) do

end
end
```

### Erlang Solution:

```
-spec categorize_box(Length :: integer(), Width :: integer(), Height ::
integer(), Mass :: integer()) -> unicode:unicode_binary().
categorize_box(Length, Width, Height, Mass) ->
.
```

### Racket Solution:

```
(define/contract (categorize-box length width height mass)
(-> exact-integer? exact-integer? exact-integer? exact-integer? string?)
)
```