# Problem 2103: Rings and Rods

## Problem Information

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

## Problem Description

There are

n

rings and each ring is either red, green, or blue. The rings are distributed

across ten rods

labeled from

0

to

9

.

You are given a string

rings

of length

2n

that describes the

n

rings that are placed onto the rods. Every two characters in

rings

forms a

color-position pair

that is used to describe each ring where:

The

first

character of the

i

th

pair denotes the

i

th

ring's

color

(

'R'

,

'G'

,

'B'

).

The

second

character of the

i

th

pair denotes the

rod

that the

i

th

ring is placed on (

'0'

to

'9'

).

For example,

"R3G2B1"

describes

n == 3

rings: a red ring placed onto the rod labeled 3, a green ring placed onto the rod labeled 2, and a blue ring placed onto the rod labeled 1.

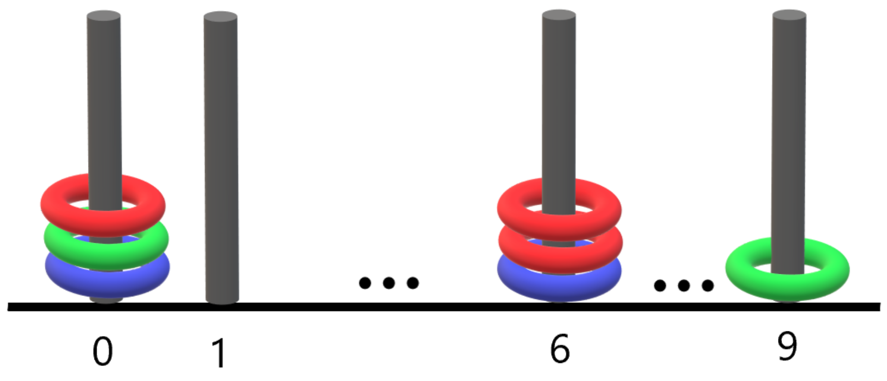
Return

the number of rods that have

all three colors

of rings on them.

Example 1:



Input:

rings = "B0B6G0R6R0R6G9"

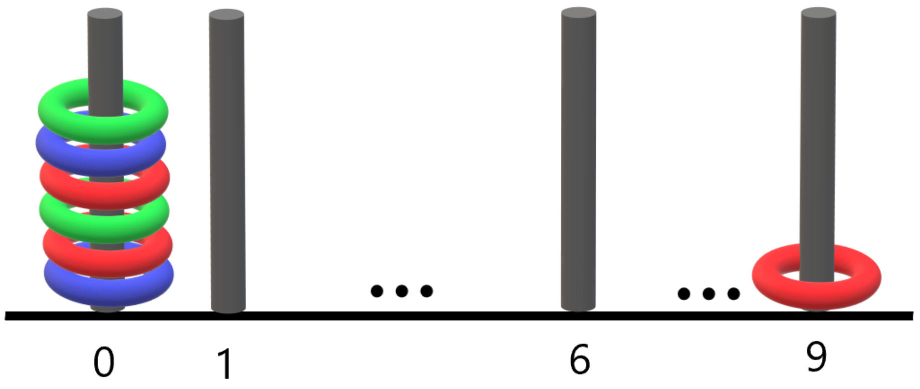
Output:

1

Explanation:

- The rod labeled 0 holds 3 rings with all colors: red, green, and blue. - The rod labeled 6 holds 3 rings, but it only has red and blue. - The rod labeled 9 holds only a green ring. Thus, the number of rods with all three colors is 1.

Example 2:



Input:

rings = "B0R0G0R9R0B0G0"

Output:

1

Explanation:

- The rod labeled 0 holds 6 rings with all colors: red, green, and blue. - The rod labeled 9 holds only a red ring. Thus, the number of rods with all three colors is 1.

Example 3:

Input:

rings = "G4"

Output:

0

Explanation:

Only one ring is given. Thus, no rods have all three colors.

Constraints:

rings.length == 2 * n

1 <= n <= 100

rings[i]

where

i

is

even

is either

'R'

,

'G'

, or

'B'

(

0-indexed

).

rings[i]

where

i

is

odd

is a digit from

'0'

to

'9'

(

0-indexed

).

## Code Snippets

**C++:**

```
class Solution {
public:
int countPoints(string rings) {

}
};
```

**Java:**

```
class Solution {
public int countPoints(String rings) {

}
}
```

**Python3:**

```
class Solution:
def countPoints(self, rings: str) -> int:
```

**Python:**

```
class Solution(object):
def countPoints(self, rings):
"""
:type rings: str
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {string} rings
* @return {number}
*/
var countPoints = function(rings) {

};
```

**TypeScript:**

```
function countPoints(rings: string): number {

};
```

**C#:**

```
public class Solution {
public int CountPoints(string rings) {

}
}
```

**C:**

```
int countPoints(char* rings) {

}
```

**Go:**

```
func countPoints(rings string) int {

}
```

**Kotlin:**

```
class Solution {
fun countPoints(rings: String): Int {

}
}
```

**Swift:**

```
class Solution {
func countPoints(_ rings: String) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn count_points(rings: String) -> i32 {

}
}
```

**Ruby:**

```
# @param {String} rings
# @return {Integer}
def count_points(rings)

end
```

**PHP:**

```
class Solution {

/**
* @param String $rings
* @return Integer
*/
function countPoints($rings) {

}
}
```

**Dart:**

```
class Solution {
int countPoints(String rings) {

}
}
```

**Scala:**

```
object Solution {
def countPoints(rings: String): Int = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec count_points(rings :: String.t) :: integer
def count_points(rings) do

end
end
```

**Erlang:**

```
-spec count_points(Rings :: unicode:unicode_binary()) -> integer().
count_points(Rings) ->
.
```

**Racket:**

```
(define/contract (count-points rings)
(-> string? exact-integer?)
)
```

## Solutions

**C++ Solution:**

```
/*
* Problem: Rings and Rods
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int countPoints(string rings) {

}
};
```

**Java Solution:**

```
/**
* Problem: Rings and Rods
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int countPoints(String rings) {
}
}
```

**Python3 Solution:**

```
"""
Problem: Rings and Rods
Difficulty: Easy
Tags: string, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def countPoints(self, rings: str) -> int:
# TODO: Implement optimized solution
pass
```

**Python Solution:**

```
class Solution(object):
def countPoints(self, rings):
"""
:type rings: str
:rtype: int
"""
```

**JavaScript Solution:**

```
/**
* Problem: Rings and Rods
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
/**
* @param {string} rings
* @return {number}
*/
var countPoints = function(rings) {

};
```

### TypeScript Solution:

```
/**
* Problem: Rings and Rods
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function countPoints(rings: string): number {

};
```

### C# Solution:

```
/*
* Problem: Rings and Rods
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int CountPoints(string rings) {

}
}
```

### C Solution:

```
/*
* Problem: Rings and Rods
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int countPoints(char* rings) {

}
```

### Go Solution:

```
// Problem: Rings and Rods
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func countPoints(rings string) int {

}
```

### Kotlin Solution:

```
class Solution {
fun countPoints(rings: String): Int {

}
}
```

### Swift Solution:

```
class Solution {
func countPoints(_ rings: String) -> Int {

}
}
```

**Rust Solution:**

```
// Problem: Rings and Rods
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn count_points(rings: String) -> i32 {

}
}
```

**Ruby Solution:**

```
# @param {String} rings
# @return {Integer}
def count_points(rings)

end
```

**PHP Solution:**

```
class Solution {

/**
* @param String $rings
* @return Integer
*/
function countPoints($rings) {

}
}
```

**Dart Solution:**

```
class Solution {
int countPoints(String rings) {

}
}
```

**Scala Solution:**

```
object Solution {
def countPoints(rings: String): Int = {

}
}
```

**Elixir Solution:**

```
defmodule Solution do
@spec count_points(rings :: String.t) :: integer
def count_points(rings) do

end
end
```

**Erlang Solution:**

```
-spec count_points(Rings :: unicode:unicode_binary()) -> integer().
count_points(Rings) ->
.
```

**Racket Solution:**

```
(define/contract (count-points rings)
(-> string? exact-integer?)
)
```