Problem 3200: Maximum Height of a Triangle

Problem Information

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

Problem Description

You are given two integers

red

and

blue

representing the count of red and blue colored balls. You have to arrange these balls to form a triangle such that the 1

st

row will have 1 ball, the 2

nd

row will have 2 balls, the 3

rd

row will have 3 balls, and so on.

All the balls in a particular row should be the

same

color, and adjacent rows should have

different

colors.

Return the

maximum

height of the triangle

that can be achieved.

Example 1:

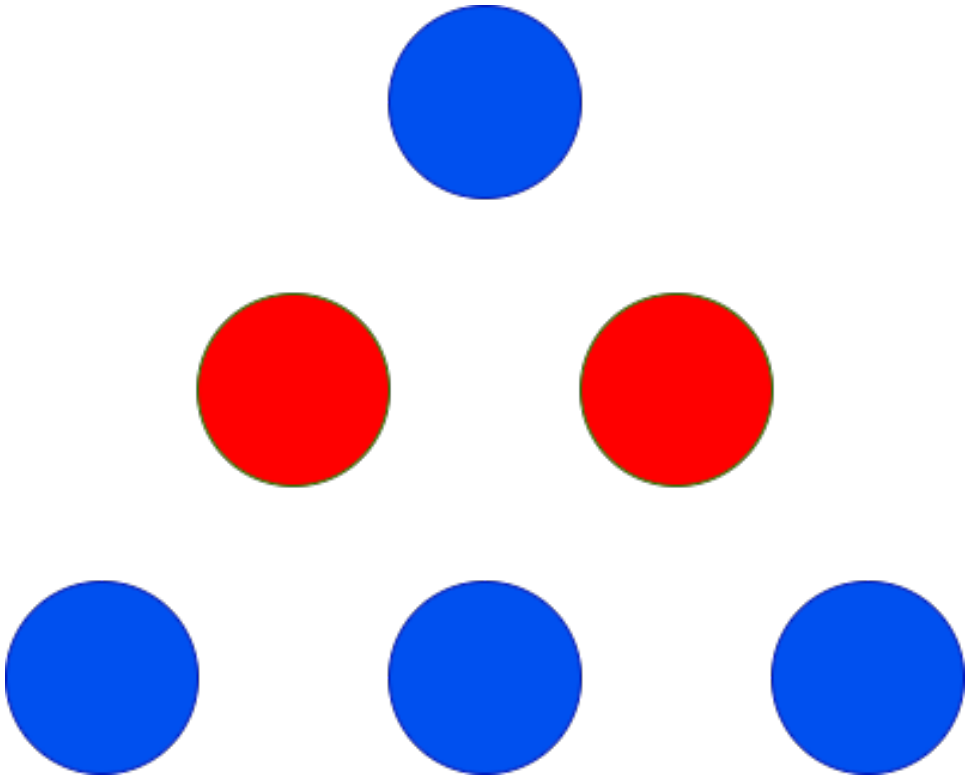
Input:

red = 2, blue = 4

Output:

3

Explanation:



The only possible arrangement is shown above.

Example 2:

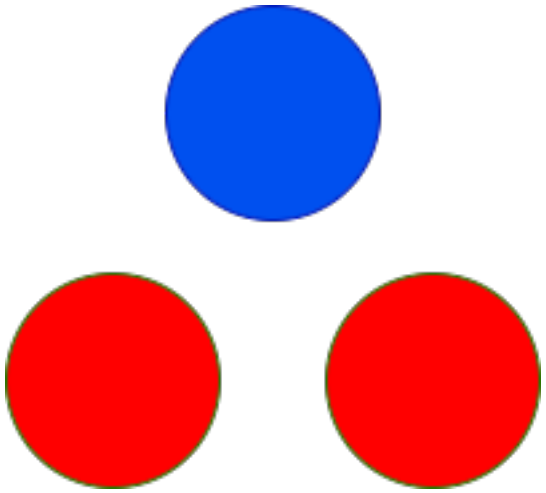
Input:

red = 2, blue = 1

Output:

2

Explanation:



The only possible arrangement is shown above.

Example 3:

Input:

red = 1, blue = 1

Output:

1

Example 4:

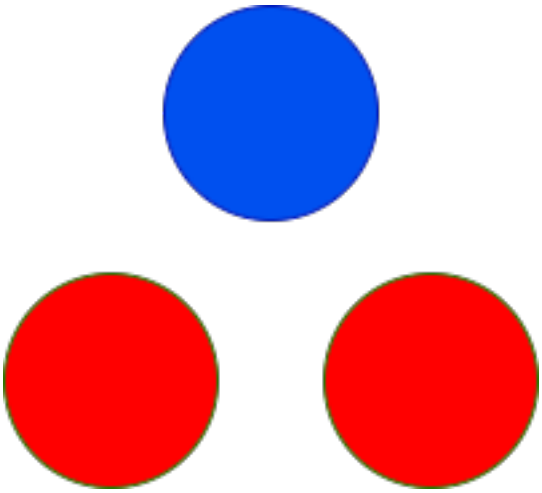
Input:

red = 10, blue = 1

Output:

2

Explanation:



The only possible arrangement is shown above.

Constraints:

1 <= red, blue <= 100

Code Snippets

C++:

```
class Solution {
public:
int maxHeightOfTriangle(int red, int blue) {

}
};
```

Java:

```
class Solution {
public int maxHeightOfTriangle(int red, int blue) {

}
}
```

Python3:

```
class Solution:
def maxHeightOfTriangle(self, red: int, blue: int) -> int:
```

Python:

```
class Solution(object):
def maxHeightOfTriangle(self, red, blue):
"""
:type red: int
:type blue: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} red
* @param {number} blue
* @return {number}
*/
var maxHeightOfTriangle = function(red, blue) {

};
```

TypeScript:

```
function maxHeightOfTriangle(red: number, blue: number): number {

};
```

C#:

```
public class Solution {
public int MaxHeightOfTriangle(int red, int blue) {

}
}
```

C:

```
int maxHeightOfTriangle(int red, int blue) {

}
```

Go:

```
func maxHeightOfTriangle(red int, blue int) int {

}
```

Kotlin:

```
class Solution {
fun maxHeightOfTriangle(red: Int, blue: Int): Int {

}
}
```

Swift:

```
class Solution {
func maxHeightOfTriangle(_ red: Int, _ blue: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_height_of_triangle(red: i32, blue: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} red
# @param {Integer} blue
# @return {Integer}
def max_height_of_triangle(red, blue)

end
```

PHP:

```
class Solution {

/**
* @param Integer $red
```

```
* @param Integer $blue
* @return Integer
*/
function maxHeightOfTriangle($red, $blue) {

}
}
```

Dart:

```
class Solution {
int maxHeightOfTriangle(int red, int blue) {

}
}
```

Scala:

```
object Solution {
def maxHeightOfTriangle(red: Int, blue: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_height_of_triangle(red :: integer, blue :: integer) :: integer
def max_height_of_triangle(red, blue) do

end
end
```

Erlang:

```
-spec max_height_of_triangle(Red :: integer(), Blue :: integer()) ->
integer().
max_height_of_triangle(Red, Blue) ->
.
```

Racket:

```
(define/contract (max-height-of-triangle red blue)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Height of a Triangle
* Difficulty: Easy
* Tags: array
*
* 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 maxHeightOfTriangle(int red, int blue) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Height of a Triangle
* Difficulty: Easy
* Tags: array
*
* 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 maxHeightOfTriangle(int red, int blue) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Maximum Height of a Triangle
Difficulty: Easy
Tags: array

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 maxHeightOfTriangle(self, red: int, blue: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxHeightOfTriangle(self, red, blue):
"""
:type red: int
:type blue: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Height of a Triangle
* Difficulty: Easy
* Tags: array
*
* 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} red
* @param {number} blue
* @return {number}
*/
var maxHeightOfTriangle = function(red, blue) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Height of a Triangle
* Difficulty: Easy
* Tags: array
*
* 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 maxHeightOfTriangle(red: number, blue: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Height of a Triangle
* Difficulty: Easy
* Tags: array
*
* 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 MaxHeightOfTriangle(int red, int blue) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Maximum Height of a Triangle
* Difficulty: Easy
* Tags: array
*
* 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 maxHeightOfTriangle(int red, int blue) {

}
```

Go Solution:

```
// Problem: Maximum Height of a Triangle
// Difficulty: Easy
// Tags: array
//
// 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 maxHeightOfTriangle(red int, blue int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxHeightOfTriangle(red: Int, blue: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxHeightOfTriangle(_ red: Int, _ blue: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Height of a Triangle
// Difficulty: Easy
// Tags: array
//
// 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 max_height_of_triangle(red: i32, blue: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} red
# @param {Integer} blue
# @return {Integer}
def max_height_of_triangle(red, blue)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $red
* @param Integer $blue
* @return Integer
*/
function maxHeightOfTriangle($red, $blue) {
```

```
}
}
```

Dart Solution:

```
class Solution {
int maxHeightOfTriangle(int red, int blue) {

}
}
```

Scala Solution:

```
object Solution {
def maxHeightOfTriangle(red: Int, blue: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_height_of_triangle(red :: integer, blue :: integer) :: integer
def max_height_of_triangle(red, blue) do

end
end
```

Erlang Solution:

```
-spec max_height_of_triangle(Red :: integer(), Blue :: integer()) ->
integer().
max_height_of_triangle(Red, Blue) ->
.
```

Racket Solution:

```
(define/contract (max-height-of-triangle red blue)
(-> exact-integer? exact-integer? exact-integer?)
)
```