Problem 390: Elimination Game

Problem Information

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

Problem Description

You have a list

arr

of all integers in the range

[1, n]

sorted in a strictly increasing order. Apply the following algorithm on

arr

:

Starting from left to right, remove the first number and every other number afterward until you reach the end of the list.

Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers.

Keep repeating the steps again, alternating left to right and right to left, until a single number remains.

Given the integer

n

, return

the last number that remains in

arr

.

Example 1:

Input:

n = 9

Output:

6

Explanation:

arr = [

1

, 2,

3

, 4,

5

, 6,

7

, 8,

9

] arr = [2,

4

, 6,

8

] arr = [

2

, 6] arr = [6]

Example 2:

Input:

n = 1

Output:

1

Constraints:

1 <= n <= 10

9

Code Snippets

C++:

```
class Solution {
public:
```

```
int lastRemaining(int n) {

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int lastRemaining(int n) {

}
```

Go:

```
func lastRemaining(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

```
(define/contract (last-remaining n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Elimination Game
* Difficulty: Medium
* Tags: math, sort
*
* 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 lastRemaining(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Elimination Game
```

```
* Difficulty: Medium
* Tags: math, sort
*
* 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 lastRemaining(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Elimination Game
Difficulty: Medium
Tags: math, sort

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 lastRemaining(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Elimination Game
* Difficulty: Medium
* Tags: math, sort
*
* 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 lastRemaining = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Elimination Game
* Difficulty: Medium
* Tags: math, sort
*
* 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 lastRemaining(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Elimination Game
* Difficulty: Medium
* Tags: math, sort
*
* 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 LastRemaining(int n) {

}
}
```

C Solution:

```
/*
* Problem: Elimination Game
* Difficulty: Medium
* Tags: math, sort
*
* 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 lastRemaining(int n) {

}
```

Go Solution:

```
// Problem: Elimination Game
// Difficulty: Medium
// Tags: math, sort
//
// 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 lastRemaining(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Elimination Game
// Difficulty: Medium
// Tags: math, sort
//
// 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 last_remaining(n: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (last-remaining n)
(-> exact-integer? exact-integer?)
)
```