Problem 1227: Airplane Seat Assignment Probability

Problem Information

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

Problem Description

n

passengers board an airplane with exactly

n

seats. The first passenger has lost the ticket and picks a seat randomly. But after that, the rest of the passengers will:

Take their own seat if it is still available, and

Pick other seats randomly when they find their seat occupied

Return

the probability that the

n

th

person gets his own seat

.

Example 1:

Input:

n = 1

Output:

1.00000

Explanation:

The first person can only get the first seat.

Example 2:

Input:

n = 2

Output:

0.50000

Explanation:

The second person has a probability of 0.5 to get the second seat (when first person gets the first seat).

Constraints:

1 <= n <= 10

5

Code Snippets

C++:

```
class Solution {
public:
double nthPersonGetsNthSeat(int n) {

}
};
```

Java:

```
class Solution {
public double nthPersonGetsNthSeat(int n) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

```
};
```

C#:

```
public class Solution {
public double NthPersonGetsNthSeat(int n) {

}
}
```

C:

```
double nthPersonGetsNthSeat(int n) {

}
```

Go:

```
func nthPersonGetsNthSeat(n int) float64 {

}
```

Kotlin:

```
class Solution {
fun nthPersonGetsNthSeat(n: Int): Double {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn nth_person_gets_nth_seat(n: i32) -> f64 {
```

```
}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
double nthPersonGetsNthSeat(int n) {

}
}
```

Scala:

```
object Solution {
def nthPersonGetsNthSeat(n: Int): Double = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec nth_person_gets_nth_seat(N :: integer()) -> float().
nth_person_gets_nth_seat(N) ->
.
```

Racket:

```
(define/contract (nth-person-gets-nth-seat n)
(-> exact-integer? flonum?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Airplane Seat Assignment Probability
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
double nthPersonGetsNthSeat(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Airplane Seat Assignment Probability
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public double nthPersonGetsNthSeat(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Airplane Seat Assignment Probability
Difficulty: Medium
Tags: dp, math

Approach: Dynamic programming with memoization or tabulation
Time Complexity: O(n * m) where n and m are problem dimensions
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def nthPersonGetsNthSeat(self, n: int) -> float:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Airplane Seat Assignment Probability
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Airplane Seat Assignment Probability
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

function nthPersonGetsNthSeat(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Airplane Seat Assignment Probability
* Difficulty: Medium
* Tags: dp, math
*
```

```
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public double NthPersonGetsNthSeat(int n) {

}
}
```

C Solution:

```
/*
* Problem: Airplane Seat Assignment Probability
* Difficulty: Medium
* Tags: dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

double nthPersonGetsNthSeat(int n) {

}
```

Go Solution:

```
// Problem: Airplane Seat Assignment Probability
// Difficulty: Medium
// Tags: dp, math
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

func nthPersonGetsNthSeat(n int) float64 {

}
```

Kotlin Solution:

```
class Solution {
fun nthPersonGetsNthSeat(n: Int): Double {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Airplane Seat Assignment Probability
// Difficulty: Medium
// Tags: dp, math
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn nth_person_gets_nth_seat(n: i32) -> f64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
double nthPersonGetsNthSeat(int n) {

}
}
```

Scala Solution:

```
object Solution {
def nthPersonGetsNthSeat(n: Int): Double = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec nth_person_gets_nth_seat(N :: integer()) -> float().
nth_person_gets_nth_seat(N) ->
.
```

Racket Solution:

```
(define/contract (nth-person-gets-nth-seat n)
(-> exact-integer? flonum?)
)
```