# Problem 837: New 21 Game

## Problem Information

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

## Problem Description

Alice plays the following game, loosely based on the card game

"21"

.

Alice starts with

0

points and draws numbers while she has less than

k

points. During each draw, she gains an integer number of points randomly from the range

[1, maxPts]

, where

maxPts

is an integer. Each draw is independent and the outcomes have equal probabilities.

Alice stops drawing numbers when she gets

k

or more points

.

Return the probability that Alice has

n

or fewer points.

Answers within

10

-5

of the actual answer are considered accepted.

Example 1:

Input:

n = 10, k = 1, maxPts = 10

Output:

1.00000

Explanation:

Alice gets a single card, then stops.

Example 2:

Input:

n = 6, k = 1, maxPts = 10

Output:

0.60000

Explanation:

Alice gets a single card, then stops. In 6 out of 10 possibilities, she is at or below 6 points.

Example 3:

Input:

n = 21, k = 17, maxPts = 10

Output:

0.73278

Constraints:

0 <= k <= n <= 10

4

1 <= maxPts <= 10

4

## Code Snippets

**C++:**

```
class Solution {
public:
double new21Game(int n, int k, int maxPts) {

}
};
```

**Java:**

```
class Solution {
public double new21Game(int n, int k, int maxPts) {

}
}
```

**Python3:**

```
class Solution:
def new21Game(self, n: int, k: int, maxPts: int) -> float:
```

**Python:**

```
class Solution(object):
def new21Game(self, n, k, maxPts):
"""
:type n: int
:type k: int
:type maxPts: int
:rtype: float
"""
```

**JavaScript:**

```
/**
* @param {number} n
* @param {number} k
* @param {number} maxPts
* @return {number}
*/
var new21Game = function(n, k, maxPts) {

};
```

**TypeScript:**

```
function new21Game(n: number, k: number, maxPts: number): number {

};
```

**C#:**

```
public class Solution {
public double New21Game(int n, int k, int maxPts) {

}
}
```

**C:**

```
double new21Game(int n, int k, int maxPts) {

}
```

**Go:**

```
func new21Game(n int, k int, maxPts int) float64 {

}
```

**Kotlin:**

```
class Solution {
fun new21Game(n: Int, k: Int, maxPts: Int): Double {

}
}
```

**Swift:**

```
class Solution {
func new21Game(_ n: Int, _ k: Int, _ maxPts: Int) -> Double {

}
}
```

**Rust:**

```
impl Solution {
pub fn new21_game(n: i32, k: i32, max_pts: i32) -> f64 {

}
}
```

**Ruby:**

```
# @param {Integer} n
# @param {Integer} k
# @param {Integer} max_pts
# @return {Float}
def new21_game(n, k, max_pts)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @param Integer $maxPts
* @return Float
*/
function new21Game($n, $k, $maxPts) {

}
}
```

**Dart:**

```
class Solution {
double new21Game(int n, int k, int maxPts) {

}
}
```

**Scala:**

```
object Solution {
def new21Game(n: Int, k: Int, maxPts: Int): Double = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec new21_game(n :: integer, k :: integer, max_pts :: integer) :: float
def new21_game(n, k, max_pts) do

end
end
```

**Erlang:**

```
-spec new21_game(N :: integer(), K :: integer(), MaxPts :: integer()) ->
float().
new21_game(N, K, MaxPts) ->
.
```

**Racket:**

```
(define/contract (new21-game n k maxPts)
(-> exact-integer? exact-integer? exact-integer? flonum?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: New 21 Game
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
double new21Game(int n, int k, int maxPts) {

}
};
```

### Java Solution:

```
/**
* Problem: New 21 Game
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public double new21Game(int n, int k, int maxPts) {

}
}
```

### Python3 Solution:

```
"""
Problem: New 21 Game
Difficulty: Medium
Tags: array, dp, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def new21Game(self, n: int, k: int, maxPts: int) -> float:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
class Solution(object):
def new21Game(self, n, k, maxPts):
"""
:type n: int
:type k: int
:type maxPts: int
```

```
:rtype: float
"""
```

### JavaScript Solution:

```
/**
* Problem: New 21 Game
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number} n
* @param {number} k
* @param {number} maxPts
* @return {number}
*/
var new21Game = function(n, k, maxPts) {

};
```

### TypeScript Solution:

```
/**
* Problem: New 21 Game
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function new21Game(n: number, k: number, maxPts: number): number {

};
```

### C# Solution:

```
/*
* Problem: New 21 Game
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public double New21Game(int n, int k, int maxPts) {

}
}
```

### C Solution:

```
/*
* Problem: New 21 Game
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

double new21Game(int n, int k, int maxPts) {

}
```

### Go Solution:

```
// Problem: New 21 Game
// Difficulty: Medium
// Tags: array, dp, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table
```

```
func new21Game(n int, k int, maxPts int) float64 {

}
```

**Kotlin Solution:**

```
class Solution {
fun new21Game(n: Int, k: Int, maxPts: Int): Double {

}
}
```

**Swift Solution:**

```
class Solution {
func new21Game(_ n: Int, _ k: Int, _ maxPts: Int) -> Double {

}
}
```

**Rust Solution:**

```
// Problem: New 21 Game
// Difficulty: Medium
// Tags: array, dp, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn new21_game(n: i32, k: i32, max_pts: i32) -> f64 {

}
}
```

**Ruby Solution:**

```
# @param {Integer} n
# @param {Integer} k
```

```
# @param {Integer} max_pts
# @return {Float}
def new21_game(n, k, max_pts)

end
```

**PHP Solution:**

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @param Integer $maxPts
* @return Float
*/
function new21Game($n, $k, $maxPts) {

}
}
```

**Dart Solution:**

```
class Solution {
double new21Game(int n, int k, int maxPts) {

}
}
```

**Scala Solution:**

```
object Solution {
def new21Game(n: Int, k: Int, maxPts: Int): Double = {

}
}
```

**Elixir Solution:**

```
defmodule Solution do
@spec new21_game(n :: integer, k :: integer, max_pts :: integer) :: float
```

```
def new21_game(n, k, max_pts) do

end
end
```

### Erlang Solution:

```
-spec new21_game(N :: integer(), K :: integer(), MaxPts :: integer()) ->
float().
new21_game(N, K, MaxPts) ->
.
```

### Racket Solution:

```
(define/contract (new21-game n k maxPts)
(-> exact-integer? exact-integer? exact-integer? flonum?)
)
```