Problem 3317: Find the Number of Possible Ways for an Event

Problem Information

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

Problem Description

You are given three integers

n

,

x

, and

y

.

An event is being held for

n

performers. When a performer arrives, they are

assigned

to one of the

x

stages. All performers assigned to the

same

stage will perform together as a band, though some stages

might

remain

empty

.

After all performances are completed, the jury will

award

each band a score in the range

[1, y]

.

Return the

total

number of possible ways the event can take place.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

Note

that two events are considered to have been held

differently

if

either

of the following conditions is satisfied:

Any

performer is

assigned

a different stage.

Any

band is

awarded

a different score.

Example 1:

Input:

n = 1, x = 2, y = 3

Output:

6

Explanation:

There are 2 ways to assign a stage to the performer.

The jury can award a score of either 1, 2, or 3 to the only band.

Example 2:

Input:

n = 5, x = 2, y = 1

Output:

32

Explanation:

Each performer will be assigned either stage 1 or stage 2.

All bands will be awarded a score of 1.

Example 3:

Input:

n = 3, x = 3, y = 4

Output:

684

Constraints:

1 <= n, x, y <= 1000

Code Snippets

C++:

```
class Solution {
public:
int numberOfWays(int n, int x, int y) {

}
};
```

Java:

```
class Solution {
public int numberOfWays(int n, int x, int y) {

}
}
```

Python3:

```
class Solution:
def numberOfWays(self, n: int, x: int, y: int) -> int:
```

Python:

```
class Solution(object):
def numberOfWays(self, n, x, y):
"""
:type n: int
:type x: int
:type y: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} x
* @param {number} y
* @return {number}
```

```
*/
var numberOfWays = function(n, x, y) {

};
```

TypeScript:

```
function numberOfWays(n: number, x: number, y: number): number {

};
```

C#:

```
public class Solution {
public int NumberOfWays(int n, int x, int y) {

}
}
```

C:

```
int numberOfWays(int n, int x, int y) {

}
```

Go:

```
func numberOfWays(n int, x int, y int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfWays(n: Int, x: Int, y: Int): Int {

}
}
```

Swift:

```
class Solution {
func numberOfWays(_ n: Int, _ x: Int, _ y: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_ways(n: i32, x: i32, y: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} x
# @param {Integer} y
# @return {Integer}
def number_of_ways(n, x, y)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $x
* @param Integer $y
* @return Integer
*/
function numberOfWays($n, $x, $y) {

}
}
```

Dart:

```
class Solution {
int numberOfWays(int n, int x, int y) {
```

```
}
}
```

Scala:

```
object Solution {
def numberOfWays(n: Int, x: Int, y: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_ways(n :: integer, x :: integer, y :: integer) :: integer
def number_of_ways(n, x, y) do

end
end
```

Erlang:

```
-spec number_of_ways(N :: integer(), X :: integer(), Y :: integer()) ->
integer().
number_of_ways(N, X, Y) ->
.
```

Racket:

```
(define/contract (number-of-ways n x y)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Number of Possible Ways for an Event
```

```
* Difficulty: Hard
* 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:
int numberOfWays(int n, int x, int y) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Number of Possible Ways for an Event
* Difficulty: Hard
* 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 int numberOfWays(int n, int x, int y) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Number of Possible Ways for an Event
Difficulty: Hard
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 numberOfWays(self, n: int, x: int, y: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfWays(self, n, x, y):
"""
:type n: int
:type x: int
:type y: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Number of Possible Ways for an Event
* Difficulty: Hard
* 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
* @param {number} x
* @param {number} y
* @return {number}
*/
var numberOfWays = function(n, x, y) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Number of Possible Ways for an Event
* Difficulty: Hard
* 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 numberOfWays(n: number, x: number, y: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Number of Possible Ways for an Event
* Difficulty: Hard
* 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 int NumberOfWays(int n, int x, int y) {

}
}
```

C Solution:

```
/*
* Problem: Find the Number of Possible Ways for an Event
* Difficulty: Hard
* 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
*/

int numberOfWays(int n, int x, int y) {

}
```

Go Solution:

```
// Problem: Find the Number of Possible Ways for an Event
// Difficulty: Hard
// 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 numberOfWays(n int, x int, y int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfWays(n: Int, x: Int, y: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfWays(_ n: Int, _ x: Int, _ y: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Number of Possible Ways for an Event
// Difficulty: Hard
// 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 number_of_ways(n: i32, x: i32, y: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} x
# @param {Integer} y
# @return {Integer}
def number_of_ways(n, x, y)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $x
* @param Integer $y
* @return Integer
*/
function numberOfWays($n, $x, $y) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfWays(int n, int x, int y) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfWays(n: Int, x: Int, y: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_ways(n :: integer, x :: integer, y :: integer) :: integer
def number_of_ways(n, x, y) do

end
end
```

Erlang Solution:

```
-spec number_of_ways(N :: integer(), X :: integer(), Y :: integer()) ->
integer().
number_of_ways(N, X, Y) ->
.
```

Racket Solution:

```
(define/contract (number-of-ways n x y)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```