Problem 552: Student Attendance Record II

Problem Information

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

Problem Description

An attendance record for a student can be represented as a string where each character signifies whether the student was absent, late, or present on that day. The record only contains the following three characters:

'A'

: Absent.

'L'

: Late.

'P'

: Present.

Any student is eligible for an attendance award if they meet

both

of the following criteria:

The student was absent (

'A'

) for

strictly

fewer than 2 days

total

.

The student was

never

late (

'L'

) for 3 or more

consecutive

days.

Given an integer

n

, return

the

number

of possible attendance records of length

n

that make a student eligible for an attendance award. The answer may be very large, so return it

modulo

10

9

+ 7

.

Example 1:

Input:

n = 2

Output:

8

Explanation:

There are 8 records with length 2 that are eligible for an award: "PP", "AP", "PA", "LP", "PL", "AL", "LA", "LL" Only "AA" is not eligible because there are 2 absences (there need to be fewer than 2).

Example 2:

Input:

n = 1

Output:

3

Example 3:

Input:

n = 10101

Output:

183236316

Constraints:

1 <= n <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int checkRecord(int n) {

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int checkRecord(int n) {

}
```

Go:

```
func checkRecord(n int) int {
```

```
}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
```

```
function checkRecord($n) {

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (check-record n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Student Attendance Record II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int checkRecord(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Student Attendance Record II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int checkRecord(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Student Attendance Record II
Difficulty: Hard
Tags: string, dp

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Student Attendance Record II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

```
};
```

TypeScript Solution:

```
/**
* Problem: Student Attendance Record II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function checkRecord(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Student Attendance Record II
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int CheckRecord(int n) {

}
}
```

C Solution:

```
/*
* Problem: Student Attendance Record II
* Difficulty: Hard
```

```
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int checkRecord(int n) {

}
```

Go Solution:

```
// Problem: Student Attendance Record II
// Difficulty: Hard
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func checkRecord(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Student Attendance Record II
// Difficulty: Hard
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

```
}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (check-record n)
(-> exact-integer? exact-integer?)
)
```