Problem 551: Student Attendance Record I

Problem Information

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

Problem Description

You are given a string

s

representing an attendance record for a student 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.

The 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.

Return

true

if the student is eligible for an attendance award, or

false

otherwise

.

Example 1:

Input:

s = "PPALLP"

Output:

true

Explanation:

The student has fewer than 2 absences and was never late 3 or more consecutive days.

Example 2:

Input:

s = "PPALLL"

Output:

false

Explanation:

The student was late 3 consecutive days in the last 3 days, so is not eligible for the award.

Constraints:

1 <= s.length <= 1000

s[i]

is either

'A'

,

'L'

, or

'P'

.

Code Snippets

C++:

```
class Solution {
public:
bool checkRecord(string s) {

}
};
```

Java:

```
class Solution {
public boolean checkRecord(String s) {

}
}
```

Python3:

```
class Solution:
def checkRecord(self, s: str) -> bool:
```

Python:

```
class Solution(object):
def checkRecord(self, s):
"""
:type s: str
```

```
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {boolean}
*/
var checkRecord = function(s) {

};
```

TypeScript:

```
function checkRecord(s: string): boolean {

};
```

C#:

```
public class Solution {
public bool CheckRecord(string s) {

}
}
```

C:

```
bool checkRecord(char* s) {

}
```

Go:

```
func checkRecord(s string) bool {

}
```

Kotlin:

```
class Solution {
fun checkRecord(s: String): Boolean {

}
}
```

Swift:

```
class Solution {
func checkRecord(_ s: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn check_record(s: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s
# @return {Boolean}
def check_record(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Boolean
*/
function checkRecord($s) {

}
}
```

Dart:

```
class Solution {
bool checkRecord(String s) {

}
}
```

Scala:

```
object Solution {
def checkRecord(s: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec check_record(s :: String.t) :: boolean
def check_record(s) do

end
end
```

Erlang:

```
-spec check_record(S :: unicode:unicode_binary()) -> boolean().
check_record(S) ->
.
```

Racket:

```
(define/contract (check-record s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Student Attendance Record I
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
bool checkRecord(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Student Attendance Record I
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public boolean checkRecord(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Student Attendance Record I
Difficulty: Easy
Tags: string
```

```
Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def checkRecord(self, s: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def checkRecord(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Student Attendance Record I
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} s
* @return {boolean}
*/
var checkRecord = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Student Attendance Record I
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function checkRecord(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Student Attendance Record I
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public bool CheckRecord(string s) {

}
}
```

C Solution:

```
/*
* Problem: Student Attendance Record I
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

bool checkRecord(char* s) {

}
```

Go Solution:

```
// Problem: Student Attendance Record I
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func checkRecord(s string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun checkRecord(s: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func checkRecord(_ s: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Student Attendance Record I
// Difficulty: Easy
// Tags: string
```

```
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn check_record(s: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Boolean}
def check_record(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Boolean
*/
function checkRecord($s) {

}
}
```

Dart Solution:

```
class Solution {
bool checkRecord(String s) {

}
}
```

Scala Solution:

```
object Solution {
def checkRecord(s: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec check_record(s :: String.t) :: boolean
def check_record(s) do

end
end
```

Erlang Solution:

```
-spec check_record(S :: unicode:unicode_binary()) -> boolean().
check_record(S) ->
.
```

Racket Solution:

```
(define/contract (check-record s)
(-> string? boolean?)
)
```