Problem 1904: The Number of Full Rounds You Have Played

Problem Information

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

Problem Description

You are participating in an online chess tournament. There is a chess round that starts every

15

minutes. The first round of the day starts at

00:00

, and after every

15

minutes, a new round starts.

For example, the second round starts at

00:15

, the fourth round starts at

00:45

, and the seventh round starts at

01:30

.

You are given two strings

loginTime

and

logoutTime

where:

loginTime

is the time you will login to the game, and

logoutTime

is the time you will logout from the game.

If

logoutTime

is

earlier

than

loginTime

, this means you have played from

loginTime

to midnight and from midnight to

logoutTime

.

Return

the number of full chess rounds you have played in the tournament

.

Note:

All the given times follow the 24-hour clock. That means the first round of the day starts at

00:00

and the last round of the day starts at

23:45

.

Example 1:

Input:

loginTime = "09:31", logoutTime = "10:14"

Output:

1

Explanation:

You played one full round from 09:45 to 10:00. You did not play the full round from 09:30 to 09:45 because you logged in at 09:31 after it began. You did not play the full round from 10:00 to 10:15 because you logged out at 10:14 before it ended.

Example 2:

Input:

loginTime = "21:30", logoutTime = "03:00"

Output:

22

Explanation:

You played 10 full rounds from 21:30 to 00:00 and 12 full rounds from 00:00 to 03:00. 10 + 12 = 22.

Constraints:

loginTime

and

logoutTime

are in the format

hh:mm

.

00 <= hh <= 23

00 <= mm <= 59

loginTime

and

logoutTime

are not equal.

Code Snippets

C++:

```
class Solution {
public:
int numberOfRounds(string loginTime, string logoutTime) {

}
};
```

Java:

```
class Solution {
public int numberOfRounds(String loginTime, String logoutTime) {

}
}
```

Python3:

```
class Solution:
def numberOfRounds(self, loginTime: str, logoutTime: str) -> int:
```

Python:

```
class Solution(object):
def numberOfRounds(self, loginTime, logoutTime):
"""
:type loginTime: str
:type logoutTime: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} loginTime
* @param {string} logoutTime
* @return {number}
*/
```

```
var numberOfRounds = function(loginTime, logoutTime) {

};
```

TypeScript:

```
function numberOfRounds(loginTime: string, logoutTime: string): number {

};
```

C#:

```
public class Solution {
public int NumberOfRounds(string loginTime, string logoutTime) {

}
}
```

C:

```
int numberOfRounds(char* loginTime, char* logoutTime) {

}
```

Go:

```
func numberOfRounds(loginTime string, logoutTime string) int {

}
```

Kotlin:

```
class Solution {
fun numberOfRounds(loginTime: String, logoutTime: String): Int {

}
}
```

Swift:

```
class Solution {
func numberOfRounds(_ loginTime: String, _ logoutTime: String) -> Int {



}
}
```

Rust:

```
impl Solution {
pub fn number_of_rounds(login_time: String, logout_time: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} login_time
# @param {String} logout_time
# @return {Integer}
def number_of_rounds(login_time, logout_time)

end
```

PHP:

```
class Solution {

/**
* @param String $loginTime
* @param String $logoutTime
* @return Integer
*/
function numberOfRounds($loginTime, $logoutTime) {

}
}
```

Dart:

```
class Solution {
int numberOfRounds(String loginTime, String logoutTime) {

}
}
```

Scala:

```
object Solution {
def numberOfRounds(loginTime: String, logoutTime: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_rounds(login_time :: String.t, logout_time :: String.t) ::
integer
def number_of_rounds(login_time, logout_time) do

end
end
```

Erlang:

```
-spec number_of_rounds(LoginTime :: unicode:unicode_binary(), LogoutTime ::
unicode:unicode_binary()) -> integer().
number_of_rounds(LoginTime, LogoutTime) ->
.
```

Racket:

```
(define/contract (number-of-rounds loginTime logoutTime)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: The Number of Full Rounds You Have Played
* Difficulty: Medium
* Tags: string, math
*
* 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:
int numberOfRounds(string loginTime, string logoutTime) {

}
};
```

Java Solution:

```
/**
* Problem: The Number of Full Rounds You Have Played
* Difficulty: Medium
* Tags: string, math
*
* 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 int numberOfRounds(String loginTime, String logoutTime) {

}
}
```

Python3 Solution:

```
"""
Problem: The Number of Full Rounds You Have Played
Difficulty: Medium
Tags: string, math

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 numberOfRounds(self, loginTime: str, logoutTime: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberOfRounds(self, loginTime, logoutTime):
"""
:type loginTime: str
:type logoutTime: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: The Number of Full Rounds You Have Played
* Difficulty: Medium
* Tags: string, math
*
* 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} loginTime
* @param {string} logoutTime
* @return {number}
*/
var numberOfRounds = function(loginTime, logoutTime) {

};
```

TypeScript Solution:

```
/**
* Problem: The Number of Full Rounds You Have Played
* Difficulty: Medium
* Tags: string, math
*
* 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 numberOfRounds(loginTime: string, logoutTime: string): number {

};
```

C# Solution:

```
/*
* Problem: The Number of Full Rounds You Have Played
* Difficulty: Medium
* Tags: string, math
*
* 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 int NumberOfRounds(string loginTime, string logoutTime) {

}
}
```

C Solution:

```
/*
* Problem: The Number of Full Rounds You Have Played
* Difficulty: Medium
* Tags: string, math
*
* 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
*/
```

```
int numberOfRounds(char* loginTime, char* logoutTime) {

}
```

Go Solution:

```
// Problem: The Number of Full Rounds You Have Played
// Difficulty: Medium
// Tags: string, math
//
// 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 numberOfRounds(loginTime string, logoutTime string) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfRounds(loginTime: String, logoutTime: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfRounds(_ loginTime: String, _ logoutTime: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: The Number of Full Rounds You Have Played
// Difficulty: Medium
// Tags: string, math
//
// 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 number_of_rounds(login_time: String, logout_time: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} login_time
# @param {String} logout_time
# @return {Integer}
def number_of_rounds(login_time, logout_time)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $loginTime
* @param String $logoutTime
* @return Integer
*/
function numberOfRounds($loginTime, $logoutTime) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfRounds(String loginTime, String logoutTime) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfRounds(loginTime: String, logoutTime: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_rounds(login_time :: String.t, logout_time :: String.t) ::
integer
def number_of_rounds(login_time, logout_time) do

end
end
```

Erlang Solution:

```
-spec number_of_rounds(LoginTime :: unicode:unicode_binary(), LogoutTime ::
unicode:unicode_binary()) -> integer().
number_of_rounds(LoginTime, LogoutTime) ->
.
```

Racket Solution:

```
(define/contract (number-of-rounds loginTime logoutTime)
(-> string? string? exact-integer?)
)
```