Problem 2224: Minimum Number of Operations to Convert Time

Problem Information

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

Problem Description

You are given two strings

current

and

correct

representing two

24-hour times

.

24-hour times are formatted as

"HH:MM"

, where

HH

is between

00

and

23

, and

MM

is between

00

and

59

. The earliest 24-hour time is

00:00

, and the latest is

23:59

.

In one operation you can increase the time

current

by

1

,

5

,

15

, or

60

minutes. You can perform this operation

any

number of times.

Return

the

minimum number of operations

needed to convert

current

to

correct

.

Example 1:

Input:

current = "02:30", correct = "04:35"

Output:

3

Explanation:

We can convert current to correct in 3 operations as follows: - Add 60 minutes to current. current becomes "03:30". - Add 60 minutes to current. current becomes "04:30". - Add 5 minutes to current. current becomes "04:35". It can be proven that it is not possible to convert current to correct in fewer than 3 operations.

Example 2:

Input:

current = "11:00", correct = "11:01"

Output:

1

Explanation:

We only have to add one minute to current, so the minimum number of operations needed is 1.

Constraints:

current

and

correct

are in the format

"HH:MM"

current <= correct

Code Snippets

C++:

```
class Solution {
public:
int convertTime(string current, string correct) {

}
};
```

Java:

```
class Solution {
public int convertTime(String current, String correct) {

}
}
```

Python3:

```
class Solution:
def convertTime(self, current: str, correct: str) -> int:
```

Python:

```
class Solution(object):
def convertTime(self, current, correct):
"""
:type current: str
:type correct: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} current
* @param {string} correct
* @return {number}
*/
var convertTime = function(current, correct) {

};
```

```
}
```

TypeScript:

```
function convertTime(current: string, correct: string): number {


};
```

C#:

```
public class Solution {
public int ConvertTime(string current, string correct) {


}
}
```

C:

```
int convertTime(char* current, char* correct) {


}
```

Go:

```
func convertTime(current string, correct string) int {


}
```

Kotlin:

```
class Solution {
fun convertTime(current: String, correct: String): Int {


}
}
```

Swift:

```
class Solution {
func convertTime(_ current: String, _ correct: String) -> Int {


}
}
```

Rust:

```
impl Solution {
pub fn convert_time(current: String, correct: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} current
# @param {String} correct
# @return {Integer}
def convert_time(current, correct)

end
```

PHP:

```
class Solution {

/**
* @param String $current
* @param String $correct
* @return Integer
*/
function convertTime($current, $correct) {

}
}
```

Dart:

```
class Solution {
int convertTime(String current, String correct) {

}
}
```

Scala:

```
object Solution {
def convertTime(current: String, correct: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec convert_time(current :: String.t, correct :: String.t) :: integer
def convert_time(current, correct) do

end
end
```

Erlang:

```
-spec convert_time(Current :: unicode:unicode_binary(), Correct ::
unicode:unicode_binary()) -> integer().
convert_time(Current, Correct) ->
.
```

Racket:

```
(define/contract (convert-time current correct)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Operations to Convert Time
* Difficulty: Easy
* Tags: string, greedy
*
* 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 convertTime(string current, string correct) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Operations to Convert Time
* Difficulty: Easy
* Tags: string, greedy
*
* 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 convertTime(String current, String correct) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Operations to Convert Time
Difficulty: Easy
Tags: string, greedy

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 convertTime(self, current: str, correct: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def convertTime(self, current, correct):
"""
:type current: str
:type correct: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Number of Operations to Convert Time
* Difficulty: Easy
* Tags: string, greedy
*
* 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} current
* @param {string} correct
* @return {number}
*/
var convertTime = function(current, correct) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Operations to Convert Time
* Difficulty: Easy
* Tags: string, greedy
*
* 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 convertTime(current: string, correct: string): number {
};
```

C# Solution:

```
/*
* Problem: Minimum Number of Operations to Convert Time
* Difficulty: Easy
* Tags: string, greedy
*
* 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 ConvertTime(string current, string correct) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Operations to Convert Time
* Difficulty: Easy
* Tags: string, greedy
*
* 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 convertTime(char* current, char* correct) {

}
```

Go Solution:

```
// Problem: Minimum Number of Operations to Convert Time
// Difficulty: Easy
// Tags: string, greedy
//
// 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 convertTime(current string, correct string) int {

}
```

Kotlin Solution:

```
class Solution {
fun convertTime(current: String, correct: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func convertTime(_ current: String, _ correct: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Operations to Convert Time
// Difficulty: Easy
// Tags: string, greedy
//
// 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 convert_time(current: String, correct: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} current
# @param {String} correct
# @return {Integer}
def convert_time(current, correct)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $current
* @param String $correct
* @return Integer
*/
function convertTime($current, $correct) {

}
}
```

Dart Solution:

```
class Solution {
int convertTime(String current, String correct) {

}
}
```

Scala Solution:

```
object Solution {
def convertTime(current: String, correct: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec convert_time(current :: String.t, correct :: String.t) :: integer
def convert_time(current, correct) do

end
end
```

Erlang Solution:

```
-spec convert_time(Current :: unicode:unicode_binary(), Correct ::
unicode:unicode_binary()) -> integer().
convert_time(Current, Correct) ->
.
```

Racket Solution:

```
(define/contract (convert-time current correct)
(-> string? string? exact-integer?)
)
```