Codility_

CodeCheck Report: trainingYVF9DM-UET

Test Name:

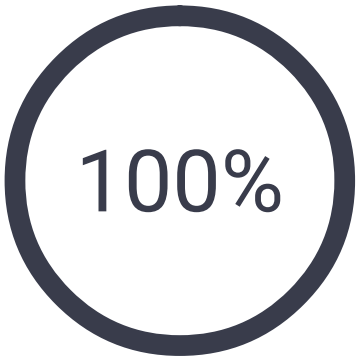
Check out Codility training tasks

Summary Timeline AI Assistant Transcript

Tasks summary

Task	Time spent	Score
ParkingBill C# ⚠	3 min	100%

Total score



Tasks Details

Elementary

1. ParkingBill

Given two strings representing times of entry and exit from a car parking lot, find the cost of the ticket according to the given billing rules.

Task Score	Correctness	Performance
100%	100%	Not assessed

Task description

You parked your car in a parking lot and want to compute the total cost of the ticket. The billing rules are as follows:

- The entrance fee of the car parking lot is 2;
- The first full or partial hour costs 3;
- Each successive full or partial hour (after the first) costs 4.

You entered the car parking lot at time E and left at time L. In this task, times are represented as strings in the format `"HH:MM"` (where `"HH"` is a two-digit number between 0 and 23, which stands for hours, and `"MM"` is a two-digit number between 0 and 59, which stands for minutes).

Write a function:

Solution

Programming language used:	C#
Total time used:	3 minutes
Effective time used:	3 minutes
Notes:	*not defined yet*

Task timeline

```
class Solution { public int solution(string E,
string L); }
```

that, given strings E and L specifying points in time in the format `"HH:MM"`, returns the total cost of the parking bill from your entry at time E to your exit at time L. You can assume that E describes a time before L on the same day.

For example, given `"10:00"` and `"13:21"` your function should return 17, because the entrance fee equals 2, the first hour costs 3 and there are two more full hours and part of a further hour, so the total cost is 2 + 3 + (3 * 4) = 17. Given `"09:42"` and `"11:42"` your function should return 9, because the entrance fee equals 2, the first hour costs 3 and the second hour costs 4, so the total cost is 2 + 3 + 4 = 9.

Assume that:

- strings E and L follow the format `"HH:MM"` strictly;
- string E describes a time before L on the same day.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

02:12:18 02:14:59

Code: 02:14:59 UTC, cs, final, score: **100** show code in pop-up

```
using System;
// you can also use other imports, for example:
// using System.Collections.Generic;

// you can write to stdout for debugging purposes,
// Console.WriteLine("this is a debug message");

class Solution {
    public int solution(string E, string L) {
        // Implement your solution here
        const int ENTRY_FEE = 2;
        const int FIRST_HOUR_FEE = 3;
        const int FOLLOWING_HOUR_FEE = 4;

        DateTime startTime = new DateTime();
        DateTime endTime = new DateTime();
        startTime = DateTime.ParseExact(E, "H:m", n
        endTime = DateTime.ParseExact(L, "H:m", nul
        int parkingFee = ENTRY_FEE + FIRST_HOUR_FEE
        var timeDifference = endTime - startTime;
        parkingFee += timeDifference.Hours * FOLLOW

        if(timeDifference.Minutes == 0)
        {
            // It's the edge case when there is a r
            parkingFee -= FOLLOWING_HOUR_FEE;
        }

        return parkingFee;
    }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

expand all	Example tests	
▶	example1 first example test	✓ OK
▶	example2 second example test	✓ OK
expand all	Correctness tests	
▶	under_ten_minutes very short parking times, answer is always 5	✓ OK
▶	random_under_hour short parking times, answer is always 5	✓ OK
▶	equal_hours_small parking for short time, always with complete hours	✓ OK
▶	random randomly generated test cases	✓ OK

▶	mixed medium and short intervals	✓ OK
▶	equal_hours_big long parking time, for complete hours	✓ OK
▶	random_big randomly generated parking times, for at least 20 hours	✓ OK
▶	maximum_result test cases giving maximum results or almost maximum results	✓ OK