Bonnet technical task

Position - Full-Stack developer

Task 1

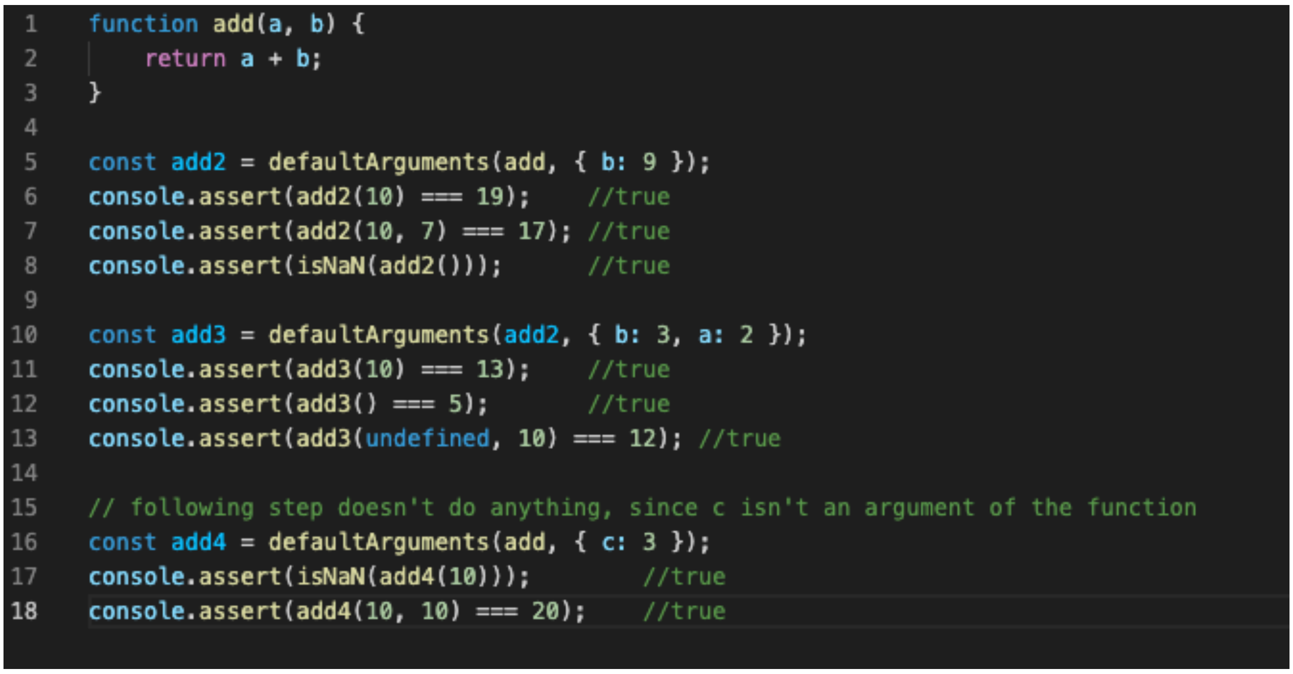
Instructions

Write a function `defaultArguments`. It takes a function as an argument, along with an object containing default values for that function's arguments, and returns another function which defaults to the right values.

Requirements

- You cannot assume that the function's arguments have any particular names.
- You should be able to call `defaultArguments` repeatedly to change the defaults.

Example:



```javascript
function add(a, b) {
    return a + b;
}

const add2 = defaultArguments(add, { b: 9 });
console.assert(add2(10) === 19);    //true
console.assert(add2(10, 7) === 17); //true
console.assert(isNaN(add2()));      //true

const add3 = defaultArguments(add2, { b: 3, a: 2 });
console.assert(add3(10) === 13);    //true
console.assert(add3() === 5);       //true
console.assert(add3(undefined, 10) === 12); //true

// following step doesn't do anything, since c isn't an argument of the function
const add4 = defaultArguments(add, { c: 3 });
console.assert(isNaN(add4(10)));        //true
console.assert(add4(10, 10) === 20);    //true
```

Task 2

Instructions

Carla “Big Mon£y” Bordania has no time to waste. She thinks, breaths and lives money. This can be seen through her motto in life: “Money is mine, and mine is money”. To maintain her highly lucrative lifestyle, Carla needs to set up lots of meetings but recently, she realised that the “set up” part of meetings often is the most time consuming and she knows very well that this time could, of course, be used to make more cash.

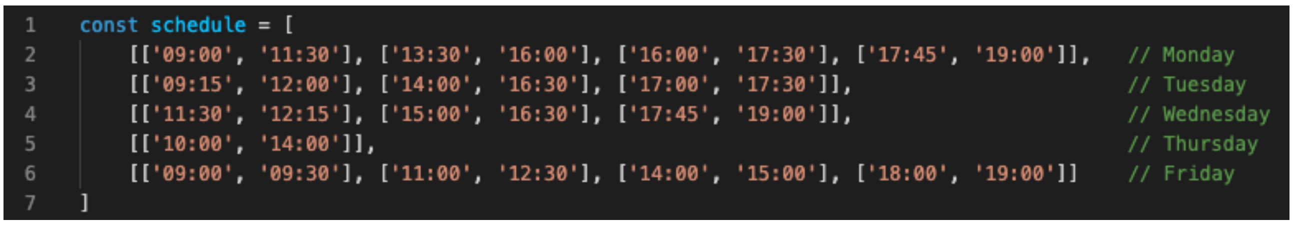
Therefore, Carla decided to hire you, a smart cookie, to create an algorithm that, when fed 2 or more schedules and a meeting duration, will automatically determine the earliest possible time for a meeting. Carla knows that with this new tool of hers, she will be spending less time on useless chit-chat and more time or valuable cash-chat.

Requirements

- All times in the calendars will be given in 24h format `hh:mm`, the result must also be in that format
- A meeting is represented by its start time (inclusively) and end time (exclusively) -> if a meeting takes place from 09:00 - 11:00, the next possible start time would be 11:00
- Carla works from 09:00 (inclusively) - 19:00 (exclusively), the appointment must start and end within that range
- If the meeting does not fit into the schedules, return `null`
- The duration of the meeting will be provided as an integer in minutes

Data Format

One schedule has the format shown below. All the schedules together will be provided to the algorithm as an array of these schedules.



```
const schedule = [
    [['09:00', '11:30'], ['13:30', '16:00'], ['16:00', '17:30'], ['17:45', '19:00']],   // Monday
    [['09:15', '12:00'], ['14:00', '16:30'], ['17:00', '17:30']],                        // Tuesday
    [['11:30', '12:15'], ['15:00', '16:30'], ['17:45', '19:00']],                        // Wednesday
    [['10:00', '14:00']],                                                                // Thursday
    [['09:00', '09:30'], ['11:00', '12:30'], ['14:00', '15:00'], ['18:00', '19:00']]    // Friday
]
```

This 3 dimensional array, contains one person’s schedule, with the top-level array containing meeting schedules for each day of the week. And within each day, meetings are represented with start and end time. For instance, in the example above, on Monday, the person who this schedule belongs to has meetings from 09:00 to 11:30, 13:30 to 16:00, 16:00 to 17:30 and 17:45 to 19:00. This means that they would be available for a meeting between 11:30 to 13:30 or 17:30 to 17:45

Results

Please write your solutions in JavaScript or TypeScript. Once you’ve come up with your solutions for both tasks, please share the code for the tasks along with at least one example for each, that can be used to test the code. This can be through GitHub or by simply sending us the files in an email.