# A. Асинхронный API из параллельной вселенной (15 баллов)

Ваш коллега-разработчик из параллельной вселенной прислал вам свою новую библиотеку для управления космическим кораблем. Т.к. космический корабль штука сложная, то и API у библиотеки довольно «развесистый», точное число методов неизвестно, документации, разумеется, нет. Зато известно, что в параллельной вселенной люди ходят по потолку, спят днём, работают ночью, а ещё используют только асинхронные функции и всегда передают callback первым аргументом. Странные ребята! У нас на Земле уже давно все на промисах пишут. Однако библиотеку нужно интегрировать в проект. Поэтому вам поступила задача написать обёртку, которая будет предоставлять тот же API, но на промисах.

## Формат ввода

Пример исходного API:

const api = {

a: {

b: {

c: callback => setTimeout(() => callback(null, ’hello’), 100)

}

},

aa: {

bb: (callback, x, y) => setTimeout(() => callback(null, x + y), 200)

}

};

## Формат вывода

Отправьте решение в виде:

/\*\*

\* @param {Object} api - исходное API

\* @returns {Object}

\*/

module.exports = function promisify(api) {

// ...

return promisedApi;

};

Пример использования:

const promisedApi = promisify(api);

promisedApi.a.b.c()

.then(res => console.log(res)); // => ’hello’

## Примечания

* обёртка должна возвращать rejected promise в случае ошибки при вызове исходного API, callback всегда принимает ошибку первым аргументом:  
  callback(error, data)
* в исходном API могут встречаться константы (числа, строки и булевые), их нужно возвращать как есть:  
  api.foo.myConst = 1;   
  promisedApi.foo.myConst === 1;
* инициализация обёртки должна быть «ленивой»: в исходном API может быть большое количество неймспейсов, и обращаться к ним нужно по мере использования.