**Как в тетрисе**

Планируя хорошо провести вечер в своём любимом ресторане на Марсе, вы сталкиваетесь с бичом звездной цивилизации - парковкой.

Заняв последнее свободное место, вы задумываетесь - неужели все должно быть *настолько* плохо? И действительно, если приглядеться, между звездолетами уйма неиспользованного места!

Администратор ресторана поражен вашим наблюдением и умоляет вас помочь: нужно реализовать программу, которая бы укомплектовала все корабли максимально эффективным образом. Самое время отплатить за гостеприимство любимом ресторану.

Условия:

* Напишите функцию parking, которая будет заполнять эту парковку таким образом, чтобы не образовывались пустоты
* Парковка на вход принимает массив уникальных кораблей
* Парковка может быть разной ширины
* Корабль имеет свой уникальный номер
* Форма корабля произвольная, ширина всегда равна ширине парковки
* Корабли имеют возможность развернуться (заехать задом или передом), чтобы заехать на парковку
* После того, как все корабли припаркуются, на парковке не должно остаться пустот
* Парковка всегда заполняется полностью (висящих пустот на любом наборе данных не будет)
* Функция должна возвращать отчёт о порядке заезда кораблей и манипуляций с их поворотами.

interface Ship {

id: number;

form: number[][];

}

interface ParkResult {

shipId: number;

position: number;

isRotated: boolean;

}

function parking(ships: Ship[]): ParkResult[] {

// code

}

**Примеры:**

*Пример 1*

const ships = [{

"id": 738,

"form": [

[1, 0],

[1, 1]

]

},

{

"id": 841,

"form": [

[1, 1],

[0, 1]

]

}];

const result = [

{

"shipId": 738,

"position": 1,

"isRotated": false

},

{

"shipId": 841,

"position": 2,

"isRotated": false

}

];

*Пример 2*

const ships = [{

"id": 443,

"form": [

[1, 0, 1],

[1, 1, 1]

]

},

{

"id": 327,

"form": [

[0, 1, 0],

[1, 1, 1],

[1, 1, 1],

[1, 1, 0],

[0, 1, 0]

]

},

{

"id": 891,

"form": [

[0, 0, 1],

[1, 0, 1],

[1, 1, 1]

]

}];

const result = [

{

"shipId": 443,

"position": 1,

"isRotated": false

},

{

"shipId": 327,

"position": 2,

"isRotated": true

},

{

"shipId": 891,

"position": 3,

"isRotated": true

}

];

*Пример 3*

const ships = [{

"id": 4892,

"form": [

[0, 0, 1],

[1, 0, 1],

[1, 1, 1],

[1, 1, 1],

[1, 1, 1],

[1, 1, 1],

[1, 1, 1],

[1, 1, 1]

]

},

{

"id": 1839,

"form": [

[1, 1, 1],

[1, 1, 1],

[1, 1, 1],

[1, 1, 1],

[1, 0, 0]

]

},

{

"id": 8183,

"form": [

[0, 1, 1],

[1, 1, 1],

[1, 1, 1],

[1, 1, 0],

[0, 1, 0]

]

}];

const result = [

{

"shipId": 4892,

"position": 1,

"isRotated": false

},

{

"shipId": 8183,

"position": 2,

"isRotated": false

},

{

"shipId": 1839,

"position": 3,

"isRotated": false

}

];

*Пример 4*

const ships = [{

"id": 1,

"form": [

[1, 0, 1],

[1, 1, 1],

[1, 1, 1]

]

},

{

"id": 2,

"form": [

[0, 0, 1],

[1, 1, 1],

[1, 1, 1],

[1, 1, 1],

[1, 1, 1]

]

},

{

"id": 3,

"form": [

[0, 1, 1],

[1, 1, 1],

[0, 1, 0]

]

}];

const result = [

{

"shipId": 1,

"position": 1,

"isRotated": false

},

{

"shipId": 3,

"position": 2,

"isRotated": false

},

{

"shipId": 2,

"position": 3,

"isRotated": true

}

];

**Примечание**

Решение должно представлять из себя Node.js модуль, экспортирующий функцию parking.

module.exports = function parking(ships) { /\* ... \*/ }