// Given this function:

function filterOutOdds() {

    var nums = Array.prototype.slice.call(arguments);

    return nums.filter(function(num) {

        return num % 2 === 0

    });

}

//refactor it to use the rest operator & an arrow function

const filterOutOdds = (...args) => args.filter(v => v % 2 === 0)

//findMin

const findMin = (...args) => Math.min(...args)

//mergeObjects

const mergeObjects = (obj1, obj2) => ({...obj1, ...obj2})

//doubleAndReturnArgs

const doubleAndReturnArgs = (arr, ...args) => [...arr, ...args.map(v => v \*2)]

//slice and dice

// remove a random element in the items array and return a new array without that item.

function removeRandom(items) {

}

const removeRandom = items => {}

    let idx = Math.floor(Math.random() \* items.length);

    return [...items.slice(o, idx), ...items.slice(idx + 1)];

/\*\* Return a new array with every item in array1 and array2. \*/

function extend(array1, array2) {

}

const extend = (array1, array2) => {

    return [...array1, ...array2];

}

/\*\* Return a new object with all the keys and values

from obj and a new key/value pair \*/

function addKeyVal(obj, key, val) {

}

const addKeyVal = (obj, key, val) => {

    let newObj = { ...obj }

    newObj[key] = val;

    return newObj;

}

/\*\* Return a new object with a key removed. \*/

function removeKey(obj, key) {

}

let newObj = { ...obj }

delete newObj[key]

return newObj;

/\*\* Combine two objects and return a new object. \*/

function combine(obj1, obj2) {

}

const combine = (obj1, obj2) => {

    return { ...obj1, ...obj2 };

}

/\*\* Return a new object with a modified key and value. \*/

function update(obj, key, val) {

}

const update = (obj, key, val) =>

    let newObj = { ...obj }

    newObj[key] = val;

    return newObj;