# Exercises: Higher-Order Functions

Problems for exercises and homework for the "Functional Programming" course from the official "Applied Programmer" curriculum.

You can check your solutions here: <https://judge.softuni.org/Contests/3588/Higher-Order-Functions-Exercises>

For the below tasks, use the **built-in array methods**.

## Duplicated Array

Your task is to write a duplicatedArray() function which finds **duplicate** **values** in a **JavaScript** **array.**

### Input

You will receive an **array of numbers** and **the searched number**.

### Output

The duplicatedArray() function should **return the new** **array** with **duplicate** **values**.

### Examples

|  |  |
| --- | --- |
| Input | Output |
| duplicatedArray([1, 2, -2, 4, 5, 4, 7, 8, 7, 7, 71, 3, 6]); | [4, 7] |
| duplicatedArray([1, 1, 2, 2]); | [1, 2] |

In Judge submit only the duplicatedArray() function.

## Filter() Implementation

Your task is to write a filterImplementation() function which **receives** an **array** of **towns** and **returns** only **towns** with **even count** of **characters.**

For the solution of this task, don't use the built-in array.prototype.filter() **method**.

### Input

You will receive an **array of strings** – **towns**.

### Output

The filterImplementation() function should **return the filtered** **array**.

### Examples

|  |  |
| --- | --- |
| Input | Output |
| filterImplementation(['Sofia', 'Pleven']); | ['Pleven'] |
| filterImplementation(['Madrid', 'Smolyan', 'Rome']); | ['Madrid', 'Rome'] |

In Judge submit only the filterImplementation() function.

## Reduce() Implementation

Your task is to write a reduceImplementation() function which **receive** an **array** of **numbers** and **return** the **sum** of the **numbers**.

For the solution of this task, don't use the built-in array.prototype.reduce() **method**.

### Input

You will receive an **array of numbers.**

### Output

The reduceImplementation() function should **return the final result** – the **sum of** **numbers**.

### Examples

|  |  |
| --- | --- |
| Input | Output |
| reduceImplementation([1, 2, 3]); | 6 |
| reduceImplementation([-1, -2, -3]); | -6 |

In Judge submit only the reduceImplementation() function.

## Leap Years

Your task is to write a leapYears() function which **finds** the **leap years** inan **array** of **years.**

### Input

You will receive an **array of years.**

### Output

The leapYears() function should **return** an **array** of **leap** **years**.

### Examples

|  |  |
| --- | --- |
| Input | Output |
| leapYears([2012, 2004, 2003]); | [2012, 2004] |
| leapYears([2002]); | [] |

In Judge submit only the leapYears() function.

## Specified Array

Your task is to write a specifiedArray() function which findsa **pair of elements (indices of the two numbers)** froma **given array** whose **sum equals** a **specific target number**.

When you find the **first** **pair** **matching** the **condition** – the **program** **ends**.

### Input

You will receive an **array of numbers** and a **target number.**

### Output

The specifiedArray() function should **return a new** **array** which includes **specific** **numbers**.

### Examples

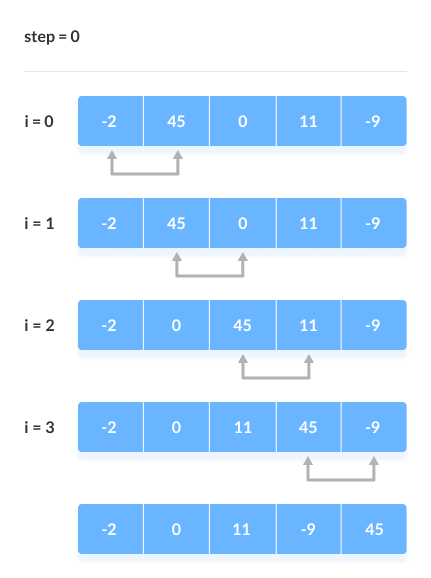
|  |  |  |
| --- | --- | --- |
| Input | Output | Explanation |
| specifiedArray([1, 2, 3], 4); | [0, 2] | 1 + 3 = 4  1 is on position 0  3 is on position 2 |
| specifiedArray([10, 20, 10, 30, 20, 60, 70], 50); | [1, 3] |  |

In Judge submit only the specifiedArray() function.

## Bubble Sort

Your task is to write а bubbleSort() function which **sorts** an array of **elements** using **Bubble** **sort.**

Bubble sort, sometimes referred to as **sinking** **sort**, is a simple **sorting** **algorithm** that repeatedly **steps** through the **list** to be **sorted**, compares each pair of adjacent items and **swaps** them if they are in the **wrong** order. The **pass** through the **list** is repeated until no **swaps** are **needed**, which **indicates** that the **list** is **sorted**.



### Input

You will receive an **array of numbers.**

### Output

The bubbleSort() function should **return the array after bubble sorting**.

### Examples

|  |  |
| --- | --- |
| Input | Output |
| bubbleSort([5, 2, 3]); | [2, 3, 5] |
| bubbleSort([3, 0, 2, 5, -1]); | [-1, 0, 2, 3, 5] |

In Judge submit only the bubbleSort() function.

## Monads and Arrays

Your task is to write a monad() function which finds an **exponent of numbers**. To solve the problem, use the array.protoype.flatMap() method.

### Input

You will receive a **matrix of numbers in format:**

[ [ number ], [ number ], … ]

### Output

The monad() function should return a **new** **array** which **includes** the **arrays** of **exponent** **numbers**.

### Examples

|  |  |
| --- | --- |
| Input | Output |
| monad([[1], [2], [3]]); | [[1], [4], [9]] |
| monad([[-1], [-2], [-3]]); | [[1], [4], [9]] |

In Judge submit only the monad() function.

## Names Order

Write a **function that orders names alphabetically**.

Graphical user interface, application, email

Description automatically generated

You will receive a **name of a student as an input**. When the [Register] button is **clicked**, you should add the **given student name** to the SoftUni Database, while **keeping** the **alphabetial order**.

For instance, if we register **David,** his name should appear in the **D** column.

Graphical user interface, text, application, email

Description automatically generated 🡪

Graphical user interface, text, application, email

Description automatically generated

If you receive **more than one name** with the **same starting letter**, you should **join all names** by **comma** **and** **space**.

Graphical user interface, application

Description automatically generated