**Задачи по JS**

**Дедлайн – суббота 19,02**

Задача 1

Вы должны реализовать функцию, которая возвращает разницу между наибольшим и наименьшим значением в списке , полученном в качестве аргумента функции. Массив, который получает функция в качестве аргумента, может содержать положительные и отрицательные числа. Если массив пуст или имеет одно значение, верните ноль. Изначально, массив будет подан в неотсортированном виде.

Пример:

arr([1, 2, 3, -4]); // вернет 7, потому что: 3 - (-4) == 7

arr([16]) => 0

Задачка 2

Напишите функцию, которая принимает в себя строку и число. Верните, в виде массива, только те слова, длинна которых превышает число.

Задача 3

В нашем неотсортированном массиве, где находятся числа от 0 до 9, потерялась одна цифра. Мы не знаем какая и нам нужно её найти.

Пример:

getMissingElement( [0,5,1,3,2,9,7,6,4]) => 8

getMissingElement( [9,2,4,5,7,0,8,6,1]) => 3

Задача 4

Напишите функцию, которая возвращает true, если первый переданный аргумент (строка) заканчивается вторым аргументом (также строкой).

Пример:

solution('abc', 'bc') => true

solution('abc', 'd') => false

Задача 5

Напишите функцию, который получает массив целых чисел и возвращает массив средних значений каждого целого числа и его последователя, если он есть.

Пример:

averages([2, -2, 2, -2, 2]), [0, 0, 0, 0]

averages([1, 3, 5, 1, -10]), [2, 4, 3, -4.5]

Задача 6

Напишите функцию, которая принимает в себя любое неотрицательное число, а возвращает максимальн возможное значение из цифр этого числа.

Пример:

function(123456789) => 987654321

function(42145) => 54421

Задача 7

Создайте функцию, которая принимает строку имени и проверяет, насколько хорошо данное имя. Предварительно загруженный словарь оценок алфавита доступен на вкладке «Код». Сложите буквы своего имени, чтобы получить общий балл.

Исходные данные:

const scores = {"A": 100, "B": 14, "C": 9, "D": 28, "E": 145, "F": 12, "G": 3,"H": 10, "I": 200, "J": 100, "K": 114, "L": 100, "M": 25,"N": 450, "O": 80, "P": 2, "Q": 12, "R": 400, "S": 113,"T": 405, "U": 11, "V": 10, "W": 10, "X": 3, "Y": 210, "Z": 23}

score <= 60: "NOT TOO GOOD"

61 <= score <= 300: "PRETTY GOOD"

301 <= score <= 599: "VERY GOOD"

score >= 600: "THE BEST"

Пример:

nameScore("MUBASHIR") ➞ "THE BEST"

nameScore("YOU") ➞ "VERY GOOD"

nameScore("MATT") ➞ "THE BEST"

nameScore("PUBG") ➞ "NOT TOO GOOD"

Задача 8

Создайте функцию, которая принимает строку и возвращает количество (количество) гласных, содержащихся в ней.

Пример:

countVowels("Celebration") ➞ 5

countVowels("Palm") ➞ 1

Задача 9

Создайте функцию, которая удалит буквы «a», «b» и «c» из данной строки и вернет измененную версию. Если данная строка не содержит «a», «b» или «c», вернуть null.

Пример:

removeABC("This might be a bit hard") ➞ "This might e  it hrd"

removeABC("hello world!") ➞ null

Задача 10

Напишите функцию JavaScript для поиска уникальных элементов из двух массивов.

Пример:

console.log(difference([1, 2, 3], [100, 2, 1, 10]));

["1", "2", "3", "10", "100"]

Задача 11

Напишите функцию, чтобы получить копию объекта, где ключи стали значениями, а значения - ключами.

Пример:

Вход - {red: "#FF0000", green: "#00FF00", white: "#FFFFFF"}

выход - {"#FF0000":"red","#00FF00":"green","#FFFFFF":"white"}

Задача 12

Ивана Иванова обокрали. Но его вещи были застрахованы на определенную сумму. Учитывая украденные вещи и ограничение страховки, верните разницу между общей стоимостью этих вещей и пределом политики.

Пример:

calculateDifference({ "baseball bat": 20 }, 5) ➞ 15

calculateDifference({ skate: 10, painting: 20 }, 19) ➞ 11

calculateDifference({ skate: 200, painting: 200, shoes: 1 }, 400) ➞ 1

*Ограничения:Объект всегда должен содержать элементы, сумма предметов всегда должна быть больше страховки.*

Задача 13

Создайте функцию вращения влево и вправо, которая возвращает все вращения строки влево и вправо.

Пример:

leftRotations("abc") ➞ ["abc", "bca", "cab"]

rightRotations("abc") ➞ ["abc", "cab", "bca"]

Задача 14

Напишите функцию, которая принимает три измерения кирпича: высоту (a), ширину (b) и глубину (c) и возвращает истину, если этот кирпич может поместиться в отверстие с шириной (w) и высотой (h).

Пример:

doesBrickFit(1, 1, 1, 1, 1) ➞ true

doesBrickFit(1, 2, 1, 1, 1) ➞ true

doesBrickFit(1, 2, 2, 1, 1) ➞ false

*Ограничения:*

*Кирпич можно повернуть любой стороной к отверстию.*

*Мы предполагаем, что кирпич подходит, если его размеры равны размерам отверстия (т.е. размер кирпича должен быть меньше или равен размеру отверстия, не строго меньше).*

*Кирпич нельзя класть под неортогональным углом.*