**1. Створити об’єкт ImmutableArray який міститиме функції для роботи з масивами, що не мутують переданий на вхід масив (повертають новий змінений):**

**○ push(arr, el)**

**○ pop(arr)**

**○ shift(arr)**

**○ unshift(arr, el)**

**○ remove(arr, index) // видалення елемента з масива по індексу**

**○ insert(arr, el, index)**

**○ swapWithPrevious(arr, index) // міняє місцями елемент з індексом index з елементом index - 1**

**○ swapWithNext(arr, index) // міняє місцями елемент з індексом index з елементом index + 1.**

var resultArray = 0;

var testArray = [1, 2, 3, 4, 5];

var ImmutableArray = {

cloneArray: function(arr) {

resultArray = arr.slice();

return resultArray;

},

push: function (arr, elem) {

resultArray = this.cloneArray(arr);

resultArray.push(elem);

return resultArray;

},

pop: function (arr) {

resultArray = this.cloneArray(arr);

resultArray.pop();

return resultArray;

},

shift: function(arr) {

resultArray = this.cloneArray(arr);

resultArray.shift();

return resultArray;

},

unshift: function(arr, elem) {

resultArray = this.cloneArray(arr);

resultArray.unshift(elem);

return resultArray;

},

remove: function(arr, index) {

resultArray = this.cloneArray(arr);

resultArray.splice(index, 1);

return resultArray;

},

insert: function(arr, elem, index) {

resultArray = this.cloneArray(arr);

resultArray.splice(index, 0, elem);

return resultArray;

},

swapWithPrevious: function(arr, index) {

resultArray = this.cloneArray(arr);

var prevElem = resultArray.splice(index - 1, 2).reverse();

resultArray.splice(index - 1, 0, ...prevElem);

return resultArray;

},

swapWithNext: function(arr, index) {

resultArray = this.cloneArray(arr);

var nextElem = resultArray.splice(index, 2).reverse();

resultArray.splice(index, 0, ...nextElem);

return resultArray;

}

}

ImmutableArray.cloneArray(testArray);

ImmutableArray.push(testArray, 6);

ImmutableArray.pop(testArray);

ImmutableArray.shift(testArray);

ImmutableArray.unshift(testArray, 0);

ImmutableArray.remove(testArray, 3);

ImmutableArray.insert(testArray, 7, 2);

ImmutableArray.swapWithPrevious(testArray, 2);

ImmutableArray.swapWithNext(testArray, 2);

**2. Написати функцію intersection(arr1, arr2) котра буде повертати перетин масивів arr1 та arr2, що складається з взятих по одному відсортованих значень, наприклад intersection([2, 1, 3, 1], [1, 3, 5]) => [1, 3].**

var testArray1 = [1, 5, 8, 2];

var testArray2 = [5, 2, 4, 1];

function intersection(arr1, arr2) {

var arr1Length = arr1.length;

var arr2Length = arr2.length;

var newArray = [];

for(var i = 0; i < arr1Length; i++) {

for(var j = 0; j < arr2Length; j++) {

if(arr1[i] === arr2[j]) {

newArray.push(arr2[j]);

newArray.sort();

};

};

};

return newArray;

}

intersection(testArray1, testArray2);

**3. Написати функцію котра буде видавати кількість голосних літер латинського алфавіту в переданій стрічці vowelCount(`Hello WORLD!`) = 3. Основна умова - використати RegExp.**

function vowelCount(str) {

var result = str.match(/[aeiouy]/gi);

return result.length;

}

vowelCount("Hello WORLD");

**4. Маючи масив, що складається з додатніх чисел, створити стрічку, елементами якої будуть йти спочатку парні числа в зростаючому порядку, а за ними непарні числа в спадаючому порядку без пробілів. Наприклад: toEvenOddString([1, 3, 2, 8, 13, 7, 4]) => `24813731` (тому що [2, 4, 8, 13, 7, 3, 1])**

var testArray = [1, 3, 2, 8, 13, 7, 4];

function toEvenOddString(arr) {

var oddNumbers = [];

var evenNumbers = [];

var resultArray = [];

var arrLength = arr.length;

for(var i = 0; i < arrLength; i++) {

if(arr[i] % 2 === 0) {

evenNumbers.push(arr[i]);

} else {

oddNumbers.push(arr[i]);

};

};

function correctSort(num1, num2) {

if(num1 > num2) return 1;

if(num1 < num2) return -1;

return 0;

};

evenNumbers.sort(correctSort);

oddNumbers.sort(correctSort).reverse();

resultArray = evenNumbers.join("") + oddNumbers.join("");

return resultArray;

}

toEvenOddString(testArray);

**6. Комплексна таска на функціональні методи масивів:**

Пререквізити

Нехай дано два масиви об’єктів-тасок з іменами і тривалостями в хвилинах

let monday = [

{

'name' : 'Write a tutorial',

'duration' : 180

},

{

'name' : 'Some web development',

'duration' : 120

}

];

let tuesday = [

{

'name' : 'Keep writing that tutorial',

'duration' : 240

},

{

'name' : 'Some more web development',

'duration' : 180

},

{

'name' : 'A whole lot of nothing',

'duration' : 240

}

];

і масив, що складається з цих двох масивів

let tasks = [monday, tuesday];

**Умова завдання**

**За допомогою тільки методів map, reduce, filter і не створюючи ніяких допоміжних масивів виконати послідовні дії:**

**1. Скомбінувати двох-вимірний масив в одно-вимірний масив усіх тасок за два дні**

let allTasks = tasks.reduce(function(sumTask, current){

return sumTask.concat(current);

}, [])

**2. Перетворити попередній масив в масив, в якому тривалість тасок перетворена з хвилин в години**

let inHours = allTasks.map(function(obj) {

obj.duration = obj.duration / 60;

return obj;

})

console.log(inHours);

**3. Вибрати тільки ті таски, тривалість яких зайняла більше 2 годин**

aboveTwo = inHours.filter(function(obj) {

if(obj.duration > 2) {

return obj;

};

})

**4. Перемножити кожну тривалість тасок на оплачуваний рейт (15$ для прикладу)**

var paidTasks = aboveTwo.map(function(obj) {

obj.duration = obj.duration \* 15;

return obj;

})

**5. Додати весь час**

var sum = paidTasks.reduce(function(sum, current) {

return sum + current.duration;

}, 0)

**6. Відформатувати отримане число у вигляді $<amount>**

var finalSum = '$' + sum

console.log(finalSum);