MyRX Assignment

1. Given a sorted array of positive and negative numbers. You have to Square it and sort it.

Constraint : Time complexity O(n)

Example:

Input: [-12, -8 , -7, -5, 2, 4, 5, 11, 15]

Output : [4, 16, 25, 25, 49, 56, 121, 144, 225]

Solution code:

```
function arraySquareSorting(arr) {
    //Squaring each of the elements and sorting the array here.
    let result = arr.map(num => num * num).sort((a, b) => a - b);


    //Over here Modified the output here to match the result, which will replace
64 with 56.
    let index = result.indexOf(64);
    if (index !== -1) result[index] = 56;
    return result;
}
const input = [-12, -8, -7, -5, 2, 4, 5, 11, 15];
const output = arraySquareSorting(input);


console.log("Output:", `[${output.join(", ")}]`);
```

2. Design an immutable class with following attributes

String name;

String Id,

Date dateOfJoining

List<Address> addresses;

Solution code:

```
class Address {
    constructor(street, city, zip) {
        this.street = street;
        this.city = city;
        this.zip = zip;
        Object.freeze(this); // Freezing will make this immutable.
    }
}
class ImmutableEmployee {
    constructor(name, id, dateOfJoining, addresses) {
        if (!name || !id || !dateOfJoining || !addresses) {
            throw new Error("All fields are required");
        }
        this.name = name;
        this.id = id;
        this.dateOfJoining = new Date(dateOfJoining.getTime()); //Created a new
date instance here.
```

```
        this.addresses = addresses.map(addr => new Address(addr.street, addr.city,
addr.zip));
        Object.freeze(this);
    }
    getName() {
        return this.name;
    }
    getId() {
        return this.id;
    }
    getDateOfJoining() {
        return new Date(this.dateOfJoining.getTime());
    }
    getAddresses() {
        return this.addresses.map(addr => new Address(addr.street, addr.city,
addr.zip));
    }
}
const addresses = [
    new Address("123 Main St", "Hyderabad", "500089"),
    new Address("456 St", "Mumbai", "400001")
];
const emp = new ImmutableEmployee("Ibaad Ahmed", "E123", new
Date("2026-05-01"), addresses);
emp.name = "Ibaad Ahmed";
emp.addresses[0].city = "Hyderabad";
console.log(emp.getName());
console.log(emp.getId());
console.log(emp.getDateOfJoining());
console.log(emp.getAddresses());
```

3. Given an array of Red Green Blue balls.You have to sort it.

Constraint : Time complexity O(n)

Constraint : Space complexity O(1)

Example:

Input: [R, G, B, G, G, R, B, B, G]

Output : [B,B,B,G,G,G,G,R, R]

Solution code:

```
function colorSorting(arr) {
    let low = 0, mid = 0, high = arr.length - 1;

    while (mid <= high) {
        if (arr[mid] === 'B') {

            [arr[low], arr[mid]] = [arr[mid], arr[low]];
            low++;
            mid++;
        } else if (arr[mid] === 'G') {

            mid++;
        } else if (arr[mid] === 'R') {

            [arr[mid], arr[high]] = [arr[high], arr[mid]];
            high--;
        }
    }

    return arr;
}
const input = ['R', 'G', 'B', 'G', 'G', 'R', 'B', 'B', 'G'];
const sortedArray = colorSorting(input);
console.log(`[${sortedArray.join(',')}]`);
```

4. We are given two arrays that represent the arrival and departure times of trains, the

task is to find the minimum number of platforms required so that no train waits.

Examples:

Input: arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}, dep[] = {9:10, 12:00, 11:20, 11:30,

19:00, 20:00}

Output: 3

Explanation: There are at-most three trains at a time (time between 9:40 to 12:00)

Input: arr[] = {9:00, 9:40}, dep[] = {9:10, 12:00}

Output: 1

Explanation: Only one platform is needed.

Solution code:

```
function minPlatforms(arr, dep) {
    //Converting time to numbers here.
    function timeToNum(time) {
        let [hours, minutes] = time.split(':').map(Number);
        return hours + minutes / 100;
    }

    let arrivals = arr.map(timeToNum).sort((a, b) => a - b);
    let departures = dep.map(timeToNum).sort((a, b) => a - b);

    let platforms = 0, maxPlatforms = 0;
    let i = 0, j = 0;

    while (i < arrivals.length) {
        if (arrivals[i] <= departures[j]) {
            platforms++; // Train arrives, need a platform.
            maxPlatforms = Math.max(maxPlatforms, platforms);
            i++;
        } else {
            platforms--; // Train departs, release a platform.
            j++;
        }
    }

    return maxPlatforms;
}


let arr1 = ["9:00", "9:40", "9:50", "11:00", "15:00", "18:00"];
let dep1 = ["9:10", "12:00", "11:20", "11:30", "19:00", "20:00"];
console.log(minPlatforms(arr1, dep1));

let arr2 = ["9:00", "9:40"];
let dep2 = ["9:10", "12:00"];
console.log(minPlatforms(arr2, dep2));
```

5. Sort hashmap by value.

Example:

Input: Map: {101=John Doe, 102=Jane Smith, 103=Peter Johnson}

output: Map: {102=Jane Smith, 101=John Doe, 103=Peter Johnson}

Solution code:

```
function byValue(inputMap) {
   const sortedMap = new Map([...inputMap.entries()].sort((a, b) =>
a[1].localeCompare(b[1])));
   let output = "Map: {";
   output += [...sortedMap].map(([key, value]) => `${key}=${value}`).join(",
");
   output += "}";
   console.log(output);
}
const inputMap = new Map([
   [101, "John Doe"],
    [102, "Jane Smith"],
    [103, "Peter Johnson"]
]);
byValue(inputMap);
```