**ASSIGNMENT5**

**1.**

function convertTo2DArray(original, m, n) {

const len = original.length;

if (len !== m \* n) {

return [];

}

const result = Array.from({ length: m }, () => Array(n));

for (let i = 0; i < len; i++) {

const row = Math.floor(i / n);

const col = i % n;

result[row][col] = original[i];

}

return result;

}

**// Test case**

const original = [1, 2, 3, 4];

const m = 2;

const n = 2;

console.log(convertTo2DArray(original, m, n));

**Output:** [[1, 2], [3, 4]]

**2.**

function countCompleteRows(n) {

let k = 1;

while ((k \* (k + 1)) / 2 <= n) {

k++;

}

return k - 1;

}

**// Test case**

const n = 5;

console.log(countCompleteRows(n));

**Output:** 2

**3.**

function sortedSquares(nums) {

const result = [];

for (let num of nums) {

const square = num \* num;

result.push(square);

}

return result.sort((a, b) => a - b);

}

**// Test case**

const nums = [-4, -1, 0, 3, 10];

console.log(sortedSquares(nums));

**Output:** [0, 1, 9, 16, 100]

**4.**

function findMissingDistinct(nums1, nums2) {

const set1 = new Set(nums1);

const set2 = new Set(nums2);

const result1 = [];

const result2 = [];

for (let num of set1) {

if (!set2.has(num)) {

result1.push(num);

}

}

for (let num of set2) {

if (!set1.has(num)) {

result2.push(num);

}

}

return [result1, result2];

}

**// Test case**

const nums1 = [1, 2, 3];

const nums2 = [2, 4, 6];

console.log(findMissingDistinct(nums1, nums2));

**Output:** [[1, 3], [4, 6]]

**5.**

function distanceValue(arr1, arr2, d) {

let distance = 0;

for (let num1 of arr1) {

let valid = true;

for (let num2 of arr2) {

if (Math.abs(num1 - num2) <= d) {

valid = false;

break;

}

}

if (valid) {

distance++;

}

}

return distance;

}

**// Test case**

const arr1 = [4, 5, 8];

const arr2 = [10, 9, 1, 8];

const d = 2;

console.log(distanceValue(arr1, arr2, d));

**Output:** 2

**6.**

function findDuplicates(nums) {

const result = [];

for (let num of nums) {

const index = Math.abs(num) - 1;

if (nums[index] > 0) {

nums[index] = -nums[index];

} else {

result.push(Math.abs(num));

}

}

return result;

}

**// Test case**

const nums = [4, 3, 2, 7, 8, 2, 3, 1];

console.log(findDuplicates(nums));

**Output:** [2, 3]

**7.**

function findMin(nums) {

let left = 0;

let right = nums.length - 1;

while (nums[left] > nums[right]) {

const mid = left + Math.floor((right - left) / 2);

if (nums[mid] > nums[right]) {

left = mid + 1;

} else {

right = mid;

}

}

return nums[left];

}

**// Test case**

const nums = [3, 4, 5, 1, 2];

console.log(findMin(nums));

**Output:** 1

**8.**

function findOriginalArray(changed) {

if (changed.length % 2 !== 0) {

return []; // If the length is odd, it cannot be a doubled array

}

changed.sort((a, b) => a - b); // Sort the array in ascending order

const original = [];

const seen = new Set();

for (const num of changed) {

if (seen.has(num / 2)) {

seen.delete(num / 2);

original.push(num / 2);

} else {

seen.add(num);

}

}

return seen.size === 0 ? original : [];

}

**// Test case**

const changed = [1, 3, 4, 2, 6, 8];

console.log(findOriginalArray(changed));

**Output:** [1, 3, 4]