**ASSIGNMENT4**

**1.**

function arraysIntersection(arr1, arr2, arr3) {

const result = [];

let i = 0

let j = 0;

let k = 0;

while (i < arr1.length && j < arr2.length && k < arr3.length) {

const num1 = arr1[i];

const num2 = arr2[j];

const num3 = arr3[k];

if (num1 === num2 && num2 === num3) {

// Common element found, add it to the result

result.push(num1);

i++;

j++;

k++;

} else if (num1 <= num2 && num1 <= num3) {

i++; // Increment pointer for arr1

} else if (num2 <= num1 && num2 <= num3) {

j++; // Increment pointer for arr2

} else {

k++; // Increment pointer for arr3

}

}

return result;

}

**// Test case**

const arr1 = [1, 2, 3, 4, 5];

const arr2 = [1, 2, 5, 7, 9];

const arr3 = [1, 3, 4, 5, 8];

console.log(arraysIntersection(arr1, arr2, arr3));

**Output:** [1, 5]

**2.**

function findMissingDistinct(nums1, nums2) {

const set1 = new Set(nums1);

const set2 = new Set(nums2);

const answer = [[], []];

for (const num of nums1) {

if (!set2.has(num)) {

answer[0].push(num);

}

}

for (const num of nums2) {

if (!set1.has(num)) {

answer[1].push(num);

}

}

return answer;

}

**// Test case**

const nums1 = [1, 2, 3];

const nums2 = [2, 4, 6];

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

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

**3.**

function transpose(matrix) {

const numRows = matrix.length;

const numCols = matrix[0].length;

const transposed = [];

for (let j = 0; j < numCols; j++) {

const newRow = [];

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

newRow.push(matrix[i][j]);

}

transposed.push(newRow);

}

return transposed;

}

**// Test case**

const matrix = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

];

console.log(transpose(matrix));

**Output:** [[1, 4, 7], [2, 5, 8], [3, 6, 9]]

**4.**

function arrayPairSum(nums) {

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

let sum = 0;

for (let i = 0; i < nums.length; i += 2) {

sum += nums[i];

}

return sum;

}

**// Test case**

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

console.log(arrayPairSum(nums));

**Output:** 4

**5.**

function arrangeCoins(n) {

let row = 1; // Start with the first row

while (n >= row) {

n -= row; // Subtract the coins required for the current row

row++; // Move to the next row

}

return row - 1; // Return the number of complete rows

}

**// Test case**

const n = 5;

console.log(arrangeCoins(n));

**Output:** 2

**6**.

function sortedSquares(nums) {

const squared = [];

for (let num of nums) {

squared.push(num \* num);

}

squared.sort((a, b) => a - b);

return squared;

}

**// Test case**

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

console.log(sortedSquares(nums));

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

**7.**

function maxCount(m, n, ops) {

let minRow = m;

let minCol = n;

for (let [ai, bi] of ops) {

minRow = Math.min(minRow, ai);

minCol = Math.min(minCol, bi);

}

const maxIntegers = minRow \* minCol;

return maxIntegers;

}

**// Test case**

const m = 3;

const n = 3;

const ops = [[2, 2], [3, 3]];

console.log(maxCount(m, n, ops));

**Output:** 4

**8.**

function rearrangeArray(nums, n) {

const result = [];

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

result.push(nums[i]);

result.push(nums[i + n]);

}

return result;

}

**// Test case**

const nums = [2, 5, 1, 3, 4, 7];

const n = 3;

console.log(rearrangeArray(nums, n));

**Output:** [2, 3, 5, 4, 1, 7]