**Assignment2**

**1.**

function arrayPairSum(nums) {

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

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];

const result = arrayPairSum(nums);

console.log(result);

**2.**

function distributeCandies(candyType) {

const uniqueCandyTypes = new Set(candyType);

return Math.min(uniqueCandyTypes.size, candyType.length / 2);

}

// Test case

const candyType = [1, 1, 2, 2, 3, 3];

const result = distributeCandies(candyType);

console.log(result);

**3.**

function findLHS(nums) {

const count = new Map();

let maxLength = 0;

for (let num of nums) {

count.set(num, (count.get(num) || 0) + 1);

}

for (let [num, frequency] of count) {

if (count.has(num + 1)) {

maxLength = Math.max(maxLength, frequency + count.get(num + 1));

}

}

return maxLength;

}

// Test case

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

const result = findLHS(nums);

console.log(result);

**4.**

function canPlaceFlowers(flowerbed, n) {

let count = 0;

let i = 0;

while (i < flowerbed.length) {

if (

flowerbed[i] === 0 &&

(i === 0 || flowerbed[i - 1] === 0) &&

(i === flowerbed.length - 1 || flowerbed[i + 1] === 0)

) {

flowerbed[i] = 1;

count++;

}

if (count >= n) {

return true;

}

i++;

}

return false;

}

// Test case

const flowerbed = [1, 0, 0, 0, 1];

const n = 1;

const result = canPlaceFlowers(flowerbed, n);

console.log(result);

**5.**

function maximumProduct(nums) {

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

const n = nums.length;

const product1 = nums[n - 1] \* nums[n - 2] \* nums[n - 3];

const product2 = nums[0] \* nums[1] \* nums[n - 1];

return Math.max(product1, product2);

}

// Test case

const nums = [1, 2, 3];

const result = maximumProduct(nums);

console.log(result);

**6.**

function search(nums, target) {

let left = 0;

let right = nums.length - 1;

while (left <= right) {

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

if (nums[mid] === target) {

return mid;

} else if (nums[mid] < target) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1;

}

// Test case

const nums = [-1, 0, 3, 5, 9, 12];

const target = 9;

const result = search(nums, target);

console.log(result);

**7.**

function isMonotonic(nums) {

const n = nums.length;

let increasing = true;

let decreasing = true;

for (let i = 1; i < n; i++) {

if (nums[i] < nums[i - 1]) {

increasing = false;

}

if (nums[i] > nums[i - 1]) {

decreasing = false;

}

}

return increasing || decreasing;

}

// Test case

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

const result = isMonotonic(nums);

console.log(result);

**8.**

function minimumScore(nums, k) {

const n = nums.length;

let minVal = Math.min(...nums);

let maxVal = Math.max(...nums);

if (maxVal - minVal === 0) {

return 0;

}

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

if (nums[i] - k >= minVal) {

minVal = Math.min(nums[i] - k, minVal);

}

if (nums[i] + k <= maxVal) {

maxVal = Math.max(nums[i] + k, maxVal);

}

}

return maxVal - minVal;

}

// Test case

const nums = [1];

const k = 0;

const result = minimumScore(nums, k);

console.log(result);