**Assignment11**

**1.**

function sqrt(x) {

if (x === 0) {

return 0;

}

let left = 1;

let right = Math.floor(x / 2) + 1;

while (left <= right) {

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

const square = mid \* mid;

if (square === x) {

return mid;

} else if (square < x) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return right;

}

**Test cases**

console.log(sqrt(4));

**Output:** 2

console.log(sqrt(8));

**Output:** 2

2.

function findPeakElement(nums) {

let left = 0;

let right = nums.length - 1;

while (left < right) {

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

if (nums[mid] < nums[mid + 1]) {

left = mid + 1;

} else {

right = mid;

}

}

return left;

}

**Test cases**

console.log(findPeakElement([1, 2, 3, 1]));

**Output:** 2

console.log(findPeakElement([1, 2, 1, 3, 5, 6, 4]));

**Output:** 5

3.

function missingNumber(nums) {

const n = nums.length;

let expectedSum = (n \* (n + 1)) / 2;

let actualSum = 0;

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

actualSum += nums[i];

}

return expectedSum - actualSum;

}

**Test cases**

console.log(missingNumber([3, 0, 1]));

**Output:** 2

console.log(missingNumber([0, 1]));

**Output:** 2

console.log(missingNumber([9, 6, 4, 2, 3, 5, 7, 0, 1]));

**Output:** 8

4.

function findDuplicate(nums) {

let slow = nums[0];

let fast = nums[0];

do {

slow = nums[slow];

fast = nums[nums[fast]];

} while (slow !== fast);

let ptr1 = nums[0];

let ptr2 = slow;

while (ptr1 !== ptr2) {

ptr1 = nums[ptr1];

ptr2 = nums[ptr2];

}

return ptr1;

}

**Test cases**

console.log(findDuplicate([1, 3, 4, 2, 2]));

**Output:** 2

console.log(findDuplicate([3, 1, 3, 4, 2]));

**Output:** 3

5.

function intersection(nums1, nums2) {

const set1 = new Set(nums1);

const intersectionSet = new Set();

for (const num of nums2) {

if (set1.has(num)) {

intersectionSet.add(num);

}

}

return Array.from(intersectionSet);

}

**Test cases**

console.log(intersection([1, 2, 2, 1], [2, 2]));

**Output:** [2]

console.log(intersection([4, 9, 5], [9, 4, 9, 8, 4]));

**Output:** [9, 4]

6.

function findMin(nums) {

let left = 0;

let right = nums.length - 1;

if (nums[left] < nums[right]) {

return nums[left];

}

while (left < right) {

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

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

left = mid + 1;

} else {

right = mid;

}

}

return nums[left];

}

**Test cases**

console.log(findMin([3, 4, 5, 1, 2]));

**Output:** 1

console.log(findMin([4, 5, 6, 7, 0, 1, 2]));

**Output:** 0

console.log(findMin([11, 13, 15, 17]));

**Output:** 11

**7.**

function searchRange(nums, target) {

const findFirstPosition = (nums, target) => {

let left = 0;

let right = nums.length - 1;

let position = -1;

while (left <= right) {

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

if (nums[mid] >= target) {

right = mid - 1;

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

position = mid;

}

} else {

left = mid + 1;

}

}

return position;

};

const findLastPosition = (nums, target) => {

let left = 0;

let right = nums.length - 1;

while (left <= right) {

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

if (nums[mid] <= target) {

left = mid + 1;

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

position = mid;

}

} else {

right = mid - 1;

}

}

return position;

};

const firstPosition = findFirstPosition(nums, target);

const lastPosition = findLastPosition(nums, target);

return [firstPosition, lastPosition];

}

**Test cases**

console.log(searchRange([5, 7, 7, 8, 8, 10], 8));

**Output:** [3, 4]

console.log(searchRange([5, 7, 7, 8, 8, 10], 6));

**Output:** [-1, -1]

console.log(searchRange([], 0));

**Output:** [-1, -1]

**8.**

function intersect(nums1, nums2) {

const countMap = new Map();

const result = [];

for (const num of nums1) {

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

}

for (const num of nums2) {

if (countMap.has(num) && countMap.get(num) > 0) {

result.push(num);

countMap.set(num, countMap.get(num) - 1);

}

}

return result;

}

**Test cases**

console.log(intersect([1, 2, 2, 1], [2, 2]));

**Output:** [2, 2]

console.log(intersect([4, 9, 5], [9, 4, 9, 8, 4]));

**Output:** [4, 9]