**Assignment19**

**1.**

function mergeTwoLists(l1, l2) {

if (!l1) return l2;

if (!l2) return l1;

if (l1.val < l2.val) {

l1.next = mergeTwoLists(l1.next, l2);

return l1;

} else {

l2.next = mergeTwoLists(l1, l2.next);

return l2;

}

}

function mergeKLists(lists) {

if (lists.length === 0) return null;

if (lists.length === 1) return lists[0];

let mergedList = lists[0];

for (let i = 1; i < lists.length; i++) {

mergedList = mergeTwoLists(mergedList, lists[i]);

}

return mergedList;

}

const lists1 = [

{ val: 1, next: { val: 4, next: { val: 5, next: null } } },

{ val: 1, next: { val: 3, next: { val: 4, next: null } } },

{ val: 2, next: { val: 6, next: null } }

];

const result1 = mergeKLists(lists1);

console.log(result1);

Output: [1,1,2,3,4,4,5,6]

const lists2 = [];

const result2 = mergeKLists(lists2);

console.log(result2);

Output: null

const lists3 = [[]];

const result3 = mergeKLists(lists3);

console.log(result3);

Output: null

2.

function mergeSort(nums, start, end, counts) {

if (start >= end) {

return nums;

}

const middle = Math.floor((start + end) / 2);

const left = mergeSort(nums, start, middle, counts);

const right = mergeSort(nums, middle + 1, end, counts);

let i = start;

let j = middle + 1;

let merged = [];

while (i <= middle && j <= end) {

if (left[i] <= right[j]) {

merged.push(left[i]);

counts[left[i]] += j - (middle + 1);

i++;

} else {

merged.push(right[j]);

j++;

}

}

while (i <= middle) {

merged.push(left[i]);

counts[left[i]] += j - (middle + 1);

i++;

}

while (j <= end) {

merged.push(right[j]);

j++;

}

for (let k = start; k <= end; k++) {

nums[k] = merged[k - start];

}

return nums;

}

function countSmaller(nums) {

const n = nums.length;

const counts = Array(n).fill(0);

mergeSort(nums, 0, n - 1, counts);

return counts;

}

console.log(countSmaller([5, 2, 6, 1]));

Output: [2, 1, 1, 0]

console.log(countSmaller([-1]));

Output: [0]

console.log(countSmaller([-1, -1]));

Output: [0, 0]

**3.**

function mergeSort(nums, start, end, sortedArr) {

if (start >= end) {

return [nums[start]];

}

const middle = Math.floor((start + end) / 2);

const left = mergeSort(nums, start, middle, sortedArr);

const right = mergeSort(nums, middle + 1, end, sortedArr);

let i = 0;

let j = 0;

let merged = [];

while (i < left.length && j < right.length) {

if (left[i] <= right[j]) {

merged.push(left[i]);

i++;

} else {

merged.push(right[j]);

j++;

}

}

while (i < left.length) {

merged.push(left[i]);

i++;

}

while (j < right.length) {

merged.push(right[j]);

j++;

}

return merged;

}

function sortArray(nums) {

return mergeSort(nums, 0, nums.length - 1, []);

}

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

Output: [1, 2, 3, 5]

console.log(sortArray([5, 1, 1, 2, 0, 0]));

Output: [0, 0, 1, 1, 2, 5]

4.

function pushZeroesToEnd(arr) {

let left = 0;

let right = arr.length - 1;

while (left < right) {

if (arr[left] !== 0) {

left++;

} else {

swap(arr, left, right);

right--;

}

}

return arr;

}

function swap(arr, i, j) {

const temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

console.log(pushZeroesToEnd([1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0]));

Output: [1, 9, 8, 4, 2, 7, 6, 0, 0, 0, 0]

console.log(pushZeroesToEnd([1, 2, 0, 4, 3, 0, 5, 0]));

Output: [1, 2, 4, 3, 5, 0, 0, 0]

console.log(pushZeroesToEnd([1, 2, 0, 0, 0, 3, 6]));

Output: [1, 2, 3, 6, 0, 0, 0]

5.

function rearrangeAlternate(arr) {

let posPtr = 0;

let negPtr = 0;

while (posPtr < arr.length && negPtr < arr.length) {

while (posPtr < arr.length && arr[posPtr] < 0) {

posPtr++;

}

negPtr = posPtr + 1;

while (negPtr < arr.length && arr[negPtr] >= 0) {

negPtr++;

}

if (posPtr < arr.length && negPtr < arr.length) {

swap(arr, posPtr, negPtr);

posPtr++;

negPtr++;

}

}

return arr;

}

function swap(arr, i, j) {

const temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

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

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

console.log(rearrangeAlternate([-5, -2, 5, 2, 4, 7, 1, 8, 0, -8]));

Output: [-5, 5, -2, 2, -8, 4, 7, 1, 8, 0]

**6.**

function mergeSortedArrays(arr1, arr2) {

const len1 = arr1.length;

const len2 = arr2.length;

const arr3 = new Array(len1 + len2);

let i = 0;

let j = 0;

let k = 0;

while (i < len1 && j < len2) {

if (arr1[i] <= arr2[j]) {

arr3[k] = arr1[i];

i++;

} else {

arr3[k] = arr2[j];

j++;

}

k++;

}

while (i < len1) {

arr3[k] = arr1[i];

i++;

k++;

}

while (j < len2) {

arr3[k] = arr2[j];

j++;

k++;

}

return arr3;

}

console.log(mergeSortedArrays([1, 3, 4, 5], [2, 4, 6, 8]));

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

console.log(mergeSortedArrays([5, 8, 9], [4, 7, 8]));

Output: [4, 5, 7, 8, 8, 9]

**7.**

function intersection(nums1, nums2) {

const set1 = new Set(nums1);

const intersection = [];

for (const num of nums2) {

if (set1.has(num)) {

intersection.push(num);

set1.delete(num);

}

}

return intersection;

}

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]

**8.**

function intersect(nums1, nums2) {

const map = new Map();

const intersection = [];

for (const num of nums1) {

if (map.has(num)) {

map.set(num, map.get(num) + 1);

} else {

map.set(num, 1);

}

}

for (const num of nums2) {

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

intersection.push(num);

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

}

}

return intersection;

}

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]