**Assignment14**

**1.**

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function detectAndRemoveLoop(head) {

if (!head || !head.next) {

return;

}

let slow = head;

let fast = head;

let loopExists = false;

while (fast && fast.next) {

slow = slow.next;

fast = fast.next.next;

if (slow === fast) {

loopExists = true;

break;

}

}

if (loopExists) {

slow = head;

while (slow !== fast.next) {

slow = slow.next;

fast = fast.next;

}

fast.next = null;

}

}

function printLinkedList(head) {

let current = head;

let result = '';

while (current) {

result += current.val + ' ';

current = current.next;

}

console.log(result.trim());

}

Example 1: N = 3, value[] = {1, 3, 4}, X = 2

const head1 = new ListNode(1);

const node2 = new ListNode(3);

const node3 = new ListNode(4);

const tail1 = new ListNode(null);

head1.next = node2;

node2.next = node3;

node3.next = tail1;

tail1.next = node2;

detectAndRemoveLoop(head1);

printLinkedList(head1);

// Output: 1 3 4

Example 2: N = 4, value[] = {1, 8, 3, 4}, X = 0

const head2 = new ListNode(1);

const node5 = new ListNode(8);

const node6 = new ListNode(3);

const tail2 = new ListNode(4);

head2.next = node5;

node5.next = node6;

node6.next = tail2;

detectAndRemoveLoop(head2);

printLinkedList(head2);

// Output: 1 8 3 4

// Example 3: N = 4, value[] = {1, 2, 3, 4}, X = 1

const head3 = new ListNode(1);

const node7 = new ListNode(2);

const node8 = new ListNode(3);

const tail3 = new ListNode(4);

head3.next = node7;

node7.next = node8;

node8.next = tail3;

tail3.next = head3;

detectAndRemoveLoop(head3);

printLinkedList(head3);

// Output: 1 2 3 4

2.

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function addOne(head) {

let prev = null;

let current = head;

while (current) {

const nextNode = current.next;

current.next = prev;

prev = current;

current = nextNode;

}

let carry = 1;

current = prev;

while (current) {

current.val += carry;

carry = Math.floor(current.val / 10);

current.val %= 10;

if (carry === 0) {

break;

}

current = current.next;

}

if (carry) {

const newNode = new ListNode(carry);

current.next = newNode;

}

prev = null;

current = prev;

while (current) {

const nextNode = current.next;

current.next = prev;

prev = current;

current = nextNode;

}

return prev;

}

function printLinkedList(head) {

let current = head;

let result = '';

while (current) {

result += current.val;

current = current.next;

}

console.log(result);

}

// Example 1: Input: LinkedList: 4->5->6

const head1 = new ListNode(4);

const node1 = new ListNode(5);

const node2 = new ListNode(6);

head1.next = node1;

node1.next = node2;

const result1 = addOne(head1);

printLinkedList(result1);

Output: 457

// Example 2: Input: LinkedList: 1->2->3

const head2 = new ListNode(1);

const node3 = new ListNode(2);

const node4 = new ListNode(3);

head2.next = node3;

node3.next = node4;

const result2 = addOne(head2);

printLinkedList(result2);

Output: 124

3.

class Node {

constructor(data) {

this.data = data;

this.next = null;

this.bottom = null;

}

}

function mergeLists(head1, head2) {

if (!head1) return head2;

if (!head2) return head1;

let result = null;

if (head1.data <= head2.data) {

result = head1;

result.bottom = mergeLists(head1.bottom, head2);

} else {

result = head2;

result.bottom = mergeLists(head1, head2.bottom);

}

return result;

}

function flattenLinkedList(head) {

if (!head || !head.next) {

return head;

}

head.next = flattenLinkedList(head.next);

head = mergeLists(head, head.next);

return head;

}

function printList(head) {

let current = head;

let result = '';

while (current) {

result += current.data + '->';

current = current.bottom;

}

console.log(result.slice(0, -2));

}

Example 1:

const head1 = new Node(5);

head1.next = new Node(10);

head1.next.next = new Node(19);

head1.next.next.next = new Node(28);

head1.bottom = new Node(7);

head1.bottom.bottom = new Node(8);

head1.bottom.bottom.bottom = new Node(30);

head1.next.bottom = new Node(20);

head1.next.next.bottom = new Node(22);

head1.next.next.next.bottom = new Node(35);

head1.next.next.next.bottom.bottom = new Node(40);

head1.next.next.next.bottom.bottom.bottom = new Node(45);

head1.next.next.next.bottom.bottom.bottom.bottom = new Node(50);

const result1 = flattenLinkedList(head1);

printList(result1);

// Output: 5->7->8->10->19->20->22->28->30->35->40->45->50

// Example 2:

const head2 = new Node(5);

head2.next = new Node(10);

head2.next.next = new Node(19);

head2.next.next.next = new Node(28);

head2.bottom = new Node(7);

head2.bottom.bottom = new Node(8);

head2.bottom.bottom.bottom = new Node(30);

head2.next.bottom = new Node(22);

head2.next.next.bottom = new Node(50);

const result2 = flattenLinkedList(head2);

printList(result2);

// Output: 5->7->8->10->19->22->28->30->50

4.

class Node {

constructor(data) {

this.data = data;

this.next = null;

this.random = null;

}

}

function copyRandomList(head) {

if (!head) return null;

const map = new Map();

// Create a copy of each node and store the mapping in the map

let current = head;

while (current) {

map.set(current, new Node(current.data));

current = current.next;

}

// Set next and random pointers for the copied nodes

current = head;

while (current) {

const copiedNode = map.get(current);

copiedNode.next = map.get(current.next);

copiedNode.random = map.get(current.random);

current = current.next;

}

return map.get(head);

}

function printList(head) {

let current = head;

let result = '';

while (current) {

const randomData = current.random ? current.random.data : 'null';

result += `${current.data} (${randomData}) -> `;

current = current.next;

}

console.log(result.slice(0, -4));

}

// Example 1:

const head1 = new Node(1);

head1.next = new Node(2);

head1.next.next = new Node(3);

head1.next.next.next = new Node(4);

head1.random = head1.next;

head1.next.random = head1.next.next.next;

const copiedHead1 = copyRandomList(head1);

printList(copiedHead1);

// Output: 1 (2) -> 2 (4) -> 3 (null) -> 4 (null)

// Example 2:

const head2 = new Node(1);

head2.next = new Node(3);

head2.next.next = new Node(5);

head2.next.next.next = new Node(9);

head2.random = head2;

head2.next.next.random = head2.next.next.next;

const copiedHead2 = copyRandomList(head2);

printList(copiedHead2);

// Output: 1 (1) -> 3 (9) -> 5 (null) -> 9 (null)

5.

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function oddEvenList(head) {

if (!head || !head.next) {

return head;

}

let oddHead = head;

let evenHead = head.next;

let odd = oddHead;

let even = evenHead;

while (even && even.next) {

odd.next = even.next;

odd = odd.next;

even.next = odd.next;

even = even.next;

}

odd.next = evenHead;

return oddHead;

}

function convertArrayToList(arr) {

if (arr.length === 0) {

return null;

}

const head = new ListNode(arr[0]);

let current = head;

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

current.next = new ListNode(arr[i]);

current = current.next;

}

return head;

}

function convertListToArray(head) {

const result = [];

let current = head;

while (current) {

result.push(current.val);

current = current.next;

}

return result;

}

// Example 1:

const list1 = convertArrayToList([1, 2, 3, 4, 5]);

const reorderedList1 = oddEvenList(list1);

const result1 = convertListToArray(reorderedList1);

console.log(result1); // Output: [1, 3, 5, 2, 4]

// Example 2:

const list2 = convertArrayToList([2, 1, 3, 5, 6, 4, 7]);

const reorderedList2 = oddEvenList(list2);

const result2 = convertListToArray(reorderedList2);

console.log(result2); // Output: [2, 3, 6, 7, 1, 5, 4]

6.

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function leftShiftLinkedList(head, k) {

if (!head || !head.next || k === 0) {

return head;

}

let current = head;

let count = 1;

while (count < k && current) {

current = current.next;

count++;

}

if (!current) {

return head;

}

const kthNode = current;

while (current.next) {

current = current.next;

}

current.next = head;

head = kthNode.next;

kthNode.next = null;

return head;

}

function convertArrayToList(arr) {

if (arr.length === 0) {

return null;

}

const head = new ListNode(arr[0]);

let current = head;

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

current.next = new ListNode(arr[i]);

current = current.next;

}

return head;

}

function convertListToArray(head) {

const result = [];

let current = head;

while (current) {

result.push(current.val);

current = current.next;

}

return result;

}

// Example 1:

const list1 = convertArrayToList([2, 4, 7, 8, 9]);

const shiftedList1 = leftShiftLinkedList(list1, 3);

const result1 = convertListToArray(shiftedList1);

console.log(result1);

// Output: [8, 9, 2, 4, 7]

// Example 2:

const list2 = convertArrayToList([1, 2, 3, 4, 5, 6, 7, 8]);

const shiftedList2 = leftShiftLinkedList(list2, 4);

const result2 = convertListToArray(shiftedList2);

console.log(result2);

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

8.

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function removeZeroSumSublists(head) {

const dummy = new ListNode(0);

dummy.next = head;

let current = dummy;

let sum = 0;

const sumMap = new Map();

while (current) {

sum += current.val;

if (sumMap.has(sum)) {

let temp = sumMap.get(sum).next;

let tempSum = sum + temp.val;

while (temp !== current) {

sumMap.delete(tempSum);

temp = temp.next;

tempSum += temp.val;

}

sumMap.get(sum).next = current.next;

} else {

sumMap.set(sum, current);

}

current = current.next;

}

return dummy.next;

}

function convertArrayToList(arr) {

if (arr.length === 0) {

return null;

}

const head = new ListNode(arr[0]);

let current = head;

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

current.next = new ListNode(arr[i]);

current = current.next;

}

return head;

}

function convertListToArray(head) {

const result = [];

let current = head;

while (current) {

result.push(current.val);

current = current.next;

}

return result;

}

// Example 1:

const list1 = convertArrayToList([1, 2, -3, 3, 1]);

const result1 = removeZeroSumSublists(list1);

console.log(convertListToArray(result1));

Output: [3, 1] or [1, 2, 1]

// Example 2:

const list2 = convertArrayToList([1, 2, 3, -3, 4]);

const result2 = removeZeroSumSublists(list2);

console.log(convertListToArray(result2));

Output: [1, 2, 4]

// Example 3:

const list3 = convertArrayToList([1, 2, 3, -3, -2]);

const result3 = removeZeroSumSublists(list3);

console.log(convertListToArray(result3));

Output: [1]