**Assignment13**

**1.**

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function createNewList(list1, list2) {

if (!list1) {

return list2;

}

if (!list2) {

return list1;

}

let curr1 = list1;

let curr2 = list2;

let newList = null;

let newHead = null;

while (curr1 && curr2) {

let newNode;

if (curr1.val >= curr2.val) {

newNode = new ListNode(curr1.val);

curr1 = curr1.next;

} else {

newNode = new ListNode(curr2.val);

curr2 = curr2.next;

}

if (!newHead) {

newHead = newNode;

} else {

newList.next = newNode;

}

newList = newNode;

}

if (curr1) {

newList.next = curr1;

} else if (curr2) {

newList.next = curr2;

}

return newHead;

}

// Example: list1 = 5->2->3->8, list2 = 1->7->4->5

const list1 = new ListNode(5);

list1.next = new ListNode(2);

list1.next.next = new ListNode(3);

list1.next.next.next = new ListNode(8);

const list2 = new ListNode(1);

list2.next = new ListNode(7);

list2.next.next = new ListNode(4);

list2.next.next.next = new ListNode(5);

const newList = createNewList(list1, list2);

console.log(newList);

// Output: 5->7->4->8

// Example: list1 = 2->8->9->3, list2 = 5->3->6->4

const list1 = new ListNode(2);

list1.next = new ListNode(8);

list1.next.next = new ListNode(9);

list1.next.next.next = new ListNode(3);

const list2 = new ListNode(5);

list2.next = new ListNode(3);

list2.next.next = new ListNode(6);

list2.next.next.next = new ListNode(4);

const newList = createNewList(list1, list2);

console.log(newList);

// Output: 5->8->9->4

2.

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function removeDuplicates(head) {

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

return head;

}

let current = head;

while (current && current.next) {

if (current.val === current.next.val) {

current.next = current.next.next;

} else {

current = current.next;

}

}

return head;

}

// Example 1: LinkedList: 11->11->11->21->43->43->60

const head1 = new ListNode(11);

head1.next = new ListNode(11);

head1.next.next = new ListNode(11);

head1.next.next.next = new ListNode(21);

head1.next.next.next.next = new ListNode(43);

head1.next.next.next.next.next = new ListNode(43);

head1.next.next.next.next.next.next = new ListNode(60);

const modifiedList1 = removeDuplicates(head1);

console.log(modifiedList1);

// Output: 11->21->43->60

// Example 2: LinkedList: 10->12->12->25->25->25->34

const head2 = new ListNode(10);

head2.next = new ListNode(12);

head2.next.next = new ListNode(12);

head2.next.next.next = new ListNode(25);

head2.next.next.next.next = new ListNode(25);

head2.next.next.next.next.next = new ListNode(25);

head2.next.next.next.next.next.next = new ListNode(34);

const modifiedList2 = removeDuplicates(head2);

console.log(modifiedList2);

// Output: 10->12->25->34

3.

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function reverseKNodes(head, k) {

if (!head || k <= 1) {

return head;

}

let current = head;

let prev = null;

while (current) {

let nextNode = null;

let prevGroupTail = null;

let count = 0;

while (current && count < k) {

nextNode = current.next;

current.next = prev;

prev = current;

current = nextNode;

count++;

}

if (nextNode) {

if (prevGroupTail) {

prevGroupTail.next = prev;

} else {

head = prev;

}

prevGroupTail = head;

while (prevGroupTail.next) {

prevGroupTail = prevGroupTail.next;

}

} else {

if (prevGroupTail) {

prevGroupTail.next = prev;

}

}

}

return head;

}

// Example 1: LinkedList: 1->2->2->4->5->6->7->8, K = 4

const head1 = new ListNode(1);

head1.next = new ListNode(2);

head1.next.next = new ListNode(2);

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

head1.next.next.next.next = new ListNode(5);

head1.next.next.next.next.next = new ListNode(6);

head1.next.next.next.next.next.next = new ListNode(7);

head1.next.next.next.next.next.next.next = new ListNode(8);

const reversedList1 = reverseKNodes(head1, 4);

console.log(reversedList1);

// Output: 4->2->2->1->8->7->6->4

// Example 2: LinkedList: 1->2->2->4->5, K = 3

const head1 = new ListNode(1);

head1.next = new ListNode(2);

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

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

head1.next.next.next.next = new ListNode(5);

const reversedList1 = reverseKNodes(head1, 3);

console.log(reversedList1);

// Output: 3->2->1->5->4

4.

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function reverseAlternateKNodes(head, k) {

if (!head || k <= 1) {

return head;

}

let current = head;

for (let i = 1; i < k && current; i++) {

current = current.next;

}

if (!current) {

return head;

}

const nextHead = current.next;

current.next = null;

reverseList(head);

head.next = reverseAlternateKNodes(nextHead, k);

return current;

}

function reverseList(head) {

let prev = null;

let current = head;

while (current) {

const nextNode = current.next;

current.next = prev;

prev = current;

current = nextNode;

}

return prev;

}

// Example: LinkedList: 1->2->3->4->5->6->7->8->9->null, k = 3

const head = new ListNode(1);

head.next = new ListNode(2);

head.next.next = new ListNode(3);

head.next.next.next = new ListNode(4);

head.next.next.next.next = new ListNode(5);

head.next.next.next.next.next = new ListNode(6);

head.next.next.next.next.next.next = new ListNode(7);

head.next.next.next.next.next.next.next = new ListNode(8);

head.next.next.next.next.next.next.next.next = new ListNode(9);

const reversedList = reverseAlternateKNodes(head, 3);

console.log(reversedList);

// Output: 3->2->1->4->5->6->9->8->7->null

5.

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function deleteLastOccurrence(head, key) {

if (!head) {

return null;

}

let prev = null;

let lastMatch = null;

let current = head;

while (current) {

if (current.val === key) {

lastMatch = current;

}

current = current.next;

}

if (!lastMatch) {

return head;

}

if (lastMatch === head) {

return head.next;

}

current = head;

while (current.next !== lastMatch) {

current = current.next;

}

current.next = lastMatch.next;

return head;

}

// Example: LinkedList: 1->2->3->5->2->10, key = 2

const head = new ListNode(1);

head.next = new ListNode(2);

head.next.next = new ListNode(3);

head.next.next.next = new ListNode(5);

head.next.next.next.next = new ListNode(2);

head.next.next.next.next.next = new ListNode(10);

const updatedList = deleteLastOccurrence(head, 2);

console.log(updatedList);

// Output: 1->2->3->5->10

6.

class ListNode {

constructor(val) {

this.val = val;

this.next = null;

}

}

function mergeLists(list1, list2) {

if (!list1) {

return list2;

}

if (!list2) {

return list1;

}

const dummy = new ListNode();

let current = dummy;

while (list1 && list2) {

if (list1.val <= list2.val) {

current.next = list1;

list1 = list1.next;

} else {

current.next = list2;

list2 = list2.next;

}

current = current.next;

}

if (list1) {

current.next = list1;

}

if (list2) {

current.next = list2;

}

return dummy.next;

}

// Example: a: 5->10->15, b: 2->3->20

const list1 = new ListNode(5);

list1.next = new ListNode(10);

list1.next.next = new ListNode(15);

const list2 = new ListNode(2);

list2.next = new ListNode(3);

list2.next.next = new ListNode(20);

const mergedList = mergeLists(list1, list2);

console.log(mergedList);

// Output: 2->3->5->10->15->20

7.

class ListNode {

constructor(val) {

this.val = val;

this.prev = null;

this.next = null;

}

}

function reverseDoublyLinkedList(head) {

let current = head;

let previous = null;

let next = null;

while (current) {

next = current.next;

current.next = previous;

current.prev = next;

previous = current;

current = next;

}

return previous;

}

// Example: Original Linked list 10 8 4 2

const head = new ListNode(10);

const node2 = new ListNode(8);

const node3 = new ListNode(4);

const tail = new ListNode(2);

head.next = node2;

node2.prev = head;

node2.next = node3;

node3.prev = node2;

node3.next = tail;

tail.prev = node3;

const reversedHead = reverseDoublyLinkedList(head);

console.log(reversedHead);

// Output: 2 4 8 10

8.

class ListNode {

constructor(val) {

this.val = val;

this.prev = null;

this.next = null;

}

}

function deleteNodeFromPosition(head, position) {

if (position === 1) {

head = head.next;

if (head) {

head.prev = null;

}

return head;

}

let current = head;

let count = 1;

while (current && count < position) {

current = current.next;

count++;

}

if (!current) {

return head;

}

current.prev.next = current.next;

if (current.next) {

current.next.prev = current.prev;

}

return head;

}

// Example: LinkedList = 1 <--> 3 <--> 4

const head = new ListNode(1);

const node2 = new ListNode(3);

const tail = new ListNode(4);

head.next = node2;

node2.prev = head;

node2.next = tail;

tail.prev = node2;

const position = 3;

const modifiedHead = deleteNodeFromPosition(head, position);

console.log(modifiedHead);

// Output: 1 3

// Example: LinkedList = 1 <--> 5 <--> 2 <--> 9

const head = new ListNode(1);

const node2 = new ListNode(5);

const node3 = new ListNode(2);

const tail = new ListNode(9);

head.next = node2;

node2.prev = head;

node2.next = node3;

node3.prev = node2;

node3.next = tail;

tail.prev = node3;

const position = 1;

const modifiedHead = deleteNodeFromPosition(head, position);

console.log(modifiedHead);

Output: 5 2 9