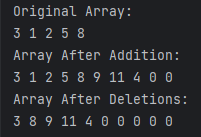
# DS-LAB 4

22K-4818

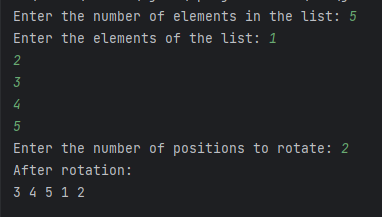
Q1.

public class Task1 {  
 public static void print(int[] array) {  
 int length = array.length;  
 for (int i = 0; i < length; i++) {  
 System.*out*.print(array[i] + " ");  
 }  
 }  
  
 public static int[] expand(int[] array) {  
 int currSize = array.length;  
 int[] temp = new int[currSize \* 2];  
 for (int i = 0; i < currSize; i++) {  
 temp[i] = array[i];  
 }  
 return temp;  
 }  
  
 public static void main(String[] args) {  
 int[] array = { 3, 1, 2, 5, 8 };  
 System.*out*.println("Original Array: ");  
 *print*(array);  
 System.*out*.println();  
  
 array = *expand*(array);  
  
 array[5] = 9;  
 array[6] = 11;  
 array[7] = 4;  
  
  
 System.*out*.println("Array After Addition: ");  
 *print*(array);  
 System.*out*.println();  
  
 int size = array.length;  
 for (int i = 0; i < size; i++) {  
 if (array[i] == 1 || array[i] == 2 || array[i] == 5) {  
 for (int j = i; j < size - 1; j++) {  
 array[j] = array[j + 1];  
 }  
 size--;  
 i--;  
 }  
 }  
 System.*out*.println("Array After Deletions: ");  
 *print*(array);  
 }  
}



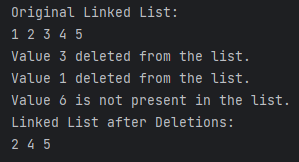
Q2.

import java.util.\*;  
  
public class Task2 {  
static class Node {  
 int data;  
 Node next;  
  
 Node(int data) {  
 this.data = data;  
 next = null;  
 }  
}  
  
 public static void print(Node head) {  
 Node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
 public static Node rotate(int k, Node head) {  
 if (k == 0 || head == null) {  
 System.*out*.println("No rotation done");  
 return head;  
 }  
  
 Node curr = head;  
 int count = 1;  
  
 while (count < k && curr != null) {  
 curr = curr.next;  
 count++;  
 }  
  
 if (curr == null) {  
 System.*out*.println("No rotation done");  
 return head;  
 }  
  
 Node kthNode = curr;  
  
 while (curr.next != null) {  
 curr = curr.next;  
 }  
  
 curr.next = head;  
 head = kthNode.next;  
 kthNode.next = null;  
  
 return head;  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Node head = null;  
 Node tail = null;  
  
 System.*out*.print("Enter the number of elements in the list: ");  
 int n = scanner.nextInt();  
  
 System.*out*.print("Enter the elements of the list: ");  
 for (int i = 0; i < n; i++) {  
 int data = scanner.nextInt();  
 Node newNode = new Node(data);  
 if (head == null) {  
 head = newNode;  
 tail = newNode;  
 } else {  
 tail.next = newNode;  
 tail = newNode;  
 }  
 }  
  
 System.*out*.print("Enter the number of positions to rotate: ");  
 int k = scanner.nextInt();  
  
 head = *rotate*(k, head);  
  
 System.*out*.println("After rotation:");  
 *print*(head);  
 }  
}



Q3.

public class Task3 {  
 class Node {  
 int data;  
 Node next;  
  
 public Node(int data) {  
 this.data = data;  
 this.next = null;  
 }  
 }  
 public static void print(Node head) {  
 Node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
 public Node delete(int value, Node head) {  
 Node curr = head;  
 Node prev = null;  
  
 while (curr != null && curr.data != value) {  
 prev = curr;  
 curr = curr.next;  
 }  
  
 if (curr == null) {  
 System.*out*.println("Value " + value + " is not present in the list.");  
 return head;  
 }  
  
 if (prev != null) {  
 prev.next = curr.next;  
 } else {  
  
 head = curr.next;  
 }  
  
 System.*out*.println("Value " + value + " deleted from the list.");  
 return head;  
 }  
  
 public static void main(String[] args) {  
 Task3 linkedlist = new Task3();  
 Node head = null;  
 Node tail = null;  
  
 int[] values = {1, 2, 3, 4, 5};  
 for (int data : values) {  
 Node newNode = linkedlist.new Node(data);  
 if (head == null) {  
 head = newNode;  
 tail = newNode;  
 } else {  
 tail.next = newNode;  
 tail = newNode;  
 }  
 }  
  
 System.*out*.println("Original Linked List:");  
 linkedlist.*print*(head);  
  
  
 head = linkedlist.delete(3, head);  
 head = linkedlist.delete(1, head);  
 head = linkedlist.delete(6, head);  
  
  
 System.*out*.println("Linked List after Deletions:");  
 linkedlist.*print*(head);  
 }  
}



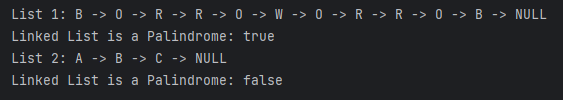
Q4.

public class Task4 {  
static class Node {  
 int data;  
 Node next;  
  
 public Node(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
static class LinkedList {  
 Node head;  
  
 public void insertAtLast(int data) {  
 Node newNode = new Node(data);  
 if (head == null) {  
 head = newNode;  
 } else {  
 Node curr = head;  
 while (curr.next != null) {  
 curr = curr.next;  
 }  
 curr.next = newNode;  
 }  
 }  
 public void modifyList() {  
 if (head == null || head.next == null) {  
 return;  
 }  
  
 Node evenHead = null;  
 Node evenTail = null;  
 Node oddHead = null;  
 Node oddTail = null;  
  
 Node current = head;  
  
 while (current != null) {  
 if (current.data % 2 == 0) {  
 if (evenHead == null) {  
 evenHead = current;  
 evenTail = current;  
 } else {  
 evenTail.next = current;  
 evenTail = current;  
 }  
 } else {  
 if (oddHead == null) {  
 oddHead = current;  
 oddTail = current;  
 } else {  
 oddTail.next = current;  
 oddTail = current;  
 }  
 }  
 current = current.next;  
 }  
  
 if (evenHead != null) {  
 evenTail.next = oddHead;  
 head = evenHead;  
 }  
 }  
 public void print() {  
 Node curr = head;  
 while (curr != null) {  
 System.*out*.print(curr.data + " -> ");  
 curr = curr.next;  
 }  
 System.*out*.println("NULL");  
 }  
}  
  
 public static void main(String[] args) {  
 LinkedList list = new LinkedList();  
 list.insertAtLast(17);  
 list.insertAtLast(15);  
 list.insertAtLast(8);  
 list.insertAtLast(12);  
 list.insertAtLast(10);  
 list.insertAtLast(5);  
 list.insertAtLast(4);  
 list.insertAtLast(1);  
 list.insertAtLast(7);  
 list.insertAtLast(6);  
  
 System.*out*.print("Original list: ");  
 list.print();  
  
 list.modifyList();  
  
 System.*out*.print("Modified list: ");  
 list.print();  
 }  
}



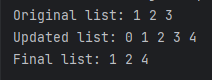
Q5.

public class Task5 {  
static class Node {  
 char data;  
 Node next;  
  
 public Node(char data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
static class LinkedList {  
 Node head;  
  
 private Node reverse(Node head) {  
 Node prev = null;  
 Node curr = head;  
 while (curr != null) {  
 Node nextTemp = curr.next;  
 curr.next = prev;  
 prev = curr;  
 curr = nextTemp;  
 }  
 return prev;  
 }  
  
 public boolean isPalindrome() {  
 if (head == null || head.next == null) {  
 return true;  
 }  
  
 Node slow = head;  
 Node fast = head;  
  
 while (fast != null && fast.next != null) {  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
  
 Node secondHalf = reverse(slow);  
 Node firstHalf = head;  
  
 while (secondHalf != null) {  
 if (firstHalf.data != secondHalf.data) {  
 return false;  
 }  
 firstHalf = firstHalf.next;  
 secondHalf = secondHalf.next;  
 }  
  
 return true;  
 }  
  
 public void print() {  
 Node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " -> ");  
 current = current.next;  
 }  
 System.*out*.println("NULL");  
 }  
  
 public void insertAtLast(int data) {  
 Node newNode = new Node((char) data);  
 if (head == null) {  
 head = newNode;  
 } else {  
 Node curr = head;  
 while (curr.next != null) {  
 curr = curr.next;  
 }  
 curr.next = newNode;  
 }  
 }  
}  
  
 public static void main(String[] args) {  
 LinkedList list1 = new LinkedList();  
 list1.insertAtLast('B');  
 list1.insertAtLast('O');  
 list1.insertAtLast('R');  
 list1.insertAtLast('R');  
 list1.insertAtLast('O');  
 list1.insertAtLast('W');  
 list1.insertAtLast('O');  
 list1.insertAtLast('R');  
 list1.insertAtLast('R');  
 list1.insertAtLast('O');  
 list1.insertAtLast('B');  
  
 System.*out*.print("List 1: ");  
 list1.print();  
  
 System.*out*.println("Linked List is a Palindrome: " + list1.isPalindrome());  
  
 LinkedList list2 = new LinkedList();  
 list2.insertAtLast('A');  
 list2.insertAtLast('B');  
 list2.insertAtLast('C');  
  
 System.*out*.print("List 2: ");  
 list2.print();  
  
 System.*out*.println("Linked List is a Palindrome: " + list2.isPalindrome());  
 }  
}



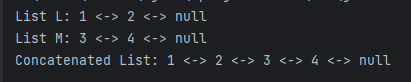
Q6.

public class Task6 {  
static class Node {  
 int data;  
 Node next;  
  
 public Node(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
static class CircularLinkedList {  
 Node head;  
  
 public void deleteNode(int data) {  
 if (head == null) {  
 System.*out*.println("List is empty");  
 return;  
 }  
 if (head.data == data) {  
 Node curr = head;  
 while (curr.next != head) {  
 curr = curr.next;  
 }  
 if (head == head.next) {  
 head = null;  
 } else {  
 head = head.next;  
 curr.next = head;  
 }  
 return;  
 }  
 Node curr = head;  
 Node prev = null;  
 while (curr.next != head) {  
 if (curr.data == data) {  
 prev.next = curr.next;  
 return;  
 }  
 prev = curr;  
 curr = curr.next;  
 }  
 if (curr.data == data) {  
 prev.next = head;  
 } else {  
 System.*out*.println("Node not found");  
 }  
 }  
  
 public void print() {  
 if (head == null) {  
 System.*out*.println("List is empty");  
 return;  
 }  
 Node curr = head;  
 do {  
 System.*out*.print(curr.data + " ");  
 curr = curr.next;  
 } while (curr != head);  
 System.*out*.println();  
 }  
  
  
 public void insertAtBeginning(int data) {  
 Node newNode = new Node(data);  
 if (head == null) {  
 head = newNode;  
 newNode.next = head;  
 } else {  
 Node curr = head;  
 while (curr.next != head) {  
 curr = curr.next;  
 }  
 newNode.next = head;  
 head = newNode;  
 curr.next = head;  
 }  
 }  
  
 public void insertAtEnd(int data) {  
 Node newNode = new Node(data);  
 if (head == null) {  
 head = newNode;  
 newNode.next = head;  
 } else {  
 Node curr = head;  
 while (curr.next != head) {  
 curr = curr.next;  
 }  
 curr.next = newNode;  
 newNode.next = head;  
 }  
 }  
  
 public void insertAtPosition(int data, int pos) {  
 Node newNode = new Node(data);  
 if (pos <= 0) {  
 System.*out*.println("Invalid position");  
 return;  
 }  
 if (pos == 1) {  
 insertAtBeginning(data);  
 return;  
 }  
 Node curr = head;  
 int count = 1;  
 while (count < pos - 1 && curr.next != head) {  
 curr = curr.next;  
 count++;  
 }  
 if (count < pos - 1) {  
 System.*out*.println("Position is out of range");  
 return;  
 }  
 newNode.next = curr.next;  
 curr.next = newNode;  
 }  
  
}  
  
 public static void main(String[] args) {  
 CircularLinkedList list = new CircularLinkedList();  
  
 list.insertAtEnd(1);  
 list.insertAtEnd(2);  
 list.insertAtEnd(3);  
  
 System.*out*.print("Original list: ");  
 list.print();  
  
 list.insertAtBeginning(0);  
 list.insertAtPosition(4, 5);  
  
 System.*out*.print("Updated list: ");  
 list.print();  
  
 list.deleteNode(0);  
 list.deleteNode(3);  
  
 System.*out*.print("Final list: ");  
 list.print();  
 }  
}



Q7.

public class Task7 {  
static class Node {  
 int data;  
 Node next;  
 Node prev;  
  
 public Node(int data) {  
 this.data = data;  
 this.next = null;  
 this.prev = null;  
 }  
}  
  
static class DoublyLinkedList {  
 Node head;  
 Node tail;  
  
 public void print() {  
 Node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " <-> ");  
 current = current.next;  
 }  
 System.*out*.println("null");  
 }  
 public void concatenate(DoublyLinkedList list2) {  
 if (list2 == null || list2.head == null) {  
 return;  
 }  
  
 if (head == null) {  
 head = list2.head;  
 tail = list2.tail;  
 } else {  
 tail.next = list2.head;  
 list2.head.prev = tail;  
 tail = list2.tail;  
 }  
 }  
}  
  
  
 public static void main(String[] args) {  
 DoublyLinkedList L = new DoublyLinkedList();  
 DoublyLinkedList M = new DoublyLinkedList();  
  
 L.head = new Node(1);  
 L.head.next = new Node(2);  
 L.tail = L.head.next;  
  
 M.head = new Node(3);  
 M.head.next = new Node(4);  
 M.tail = M.head.next;  
  
 System.*out*.print("List L: ");  
 L.print();  
 System.*out*.print("List M: ");  
 M.print();  
  
 L.concatenate(M);  
  
 System.*out*.print("Concatenated List: ");  
 L.print();  
 }  
}



Q8.

public class Task8 {  
static class Node {  
 int data;  
 Node next;  
  
 public Node(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
static class LinkedList {  
 Node head;  
  
 public void print() {  
 Node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
 public void insertAtLast(int data) {  
 Node newNode = new Node(data);  
 if (head == null) {  
 head = newNode;  
 } else {  
 Node curr = head;  
 while (curr.next != null) {  
 curr = curr.next;  
 }  
 curr.next = newNode;  
 }  
 }  
  
 public void modifyLinkedList() {  
 if (head == null || head.next == null) {  
 return;  
 }  
  
 Node curr = head;  
 Node alt = head.next;  
 Node newHead = null;  
  
 while (curr != null && alt != null) {  
 curr.next = alt.next;  
 alt.next = newHead;  
 newHead = alt;  
 alt = curr.next;  
  
 if (alt != null) {  
 curr = alt;  
 alt = curr.next;  
 }  
 }  
  
 curr.next = newHead;  
 }  
  
  
}  
  
 public static void main(String[] args) {  
 LinkedList list = new LinkedList();  
 list.insertAtLast(10);  
 list.insertAtLast(4);  
 list.insertAtLast(9);  
 list.insertAtLast(1);  
 list.insertAtLast(3);  
 list.insertAtLast(5);  
 list.insertAtLast(9);  
 list.insertAtLast(4);  
  
 System.*out*.print("Original list: ");  
 list.print();  
  
 list.modifyLinkedList();  
  
 System.*out*.print("Modified list: ");  
 list.print();  
 }  
}

