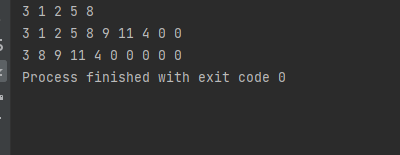
Mustafa Masood 22k-4818 Lab 4

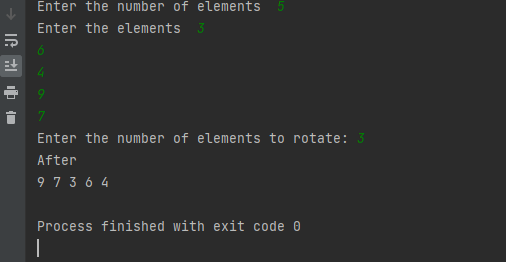
Q1:

public class q1 {  
  
 public static void print(int[] array){  
 int n=array.length;  
 for( int i= 0; i<n; i++){  
 System.*out*.print(array[i] + " ");  
 }  
 }  
  
 public static int[] growSize(int [] array)  
 {  
 int sizeofarray = array.length;  
  
 int temp[] = null;  
  
 temp = new int[array.length \* 2];  
 {  
 for (int i = 0; i < array.length; i++)  
 {  
  
 temp[i] = array[i];  
 }  
 }  
  
 array = temp;  
 sizeofarray= array.length \* 2;  
  
 return array;  
 }  
  
  
 public static void main(String[] args) {  
 int [] array = {3,1,2,5,8};  
 *print*(array);  
  
 array = *growSize*(array);  
  
 array[4+1] = 9;  
 array[4+2] = 11;  
 array[4+3] = 4;  
  
 System.*out*.println();  
 *print*(array);  
  
 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();  
 *print*(array);  
 }  
}

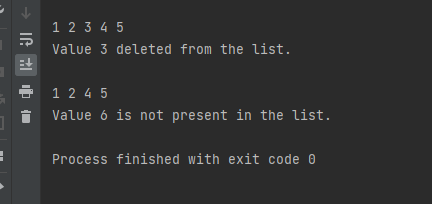


Q2:

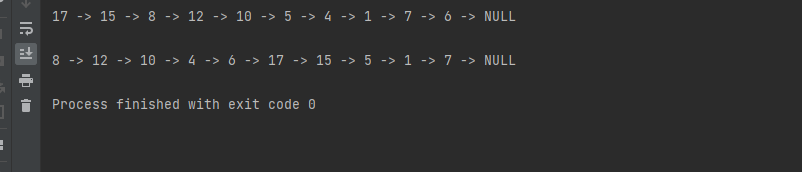
import java.util.Scanner;  
class Node {  
 int data;  
 Node next;  
  
 Node(int data) {  
 this.data = data;  
 next = null;  
 }  
}  
public class q2 {  
 public static Node rotateLinkedList(Node head, int k) {  
 if (head == null || k == 0) {  
 return head;  
 }  
  
 Node current = head;  
 int count = 1;  
  
 while (count < k && current != null) {  
 current = current.next;  
 count++;  
 }  
  
 if (current == null) {  
 return head; // k is greater than or equal to the length of the list, no rotation needed  
 }  
  
 Node kthNode = current;  
 while (current.next != null) {  
 current = current.next;  
 }  
  
 current.next = head;  
 head = kthNode.next;  
 kthNode.next = null;  
  
 return head;  
 }  
  
 public static void printList(Node head) {  
 Node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
  
 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 ");  
 int n = scanner.nextInt();  
  
 System.*out*.print("Enter the elements ");  
 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 elements to rotate: ");  
 int k = scanner.nextInt();  
  
 head = *rotateLinkedList*(head, k);  
  
 System.*out*.println("After ");  
 *printList*(head);  
  
 }  
}



Q3:  
  
class q3 {  
  
 class Node {  
 int data;  
 Node next;  
  
 public Node(int data) {  
 this.data = data;  
 this.next = null;  
 }  
 }  
  
 Node head;  
  
 public q3() {  
 this.head = null;  
 }  
  
  
 public void insert(int data) {  
 Node newNode = new Node(data);  
 if (head == null) {  
 head = newNode;  
 } else {  
 Node current = head;  
 while (current.next != null) {  
 current = current.next;  
 }  
 current.next = newNode;  
 }  
 }  
  
 public void delete(int value) {  
 if (head == null) {  
 System.*out*.println("The list is empty.");  
 return;  
 }  
  
 if (head.data == value) {  
 head = head.next;  
 return;  
 }  
  
 Node current = head;  
 Node previous = null;  
  
 while (current != null && current.data != value) {  
 previous = current;  
 current = current.next;  
 }  
  
 if (current == null) {  
 System.*out*.println("Value " + value + " is not present in the list.");  
 } else {  
 previous.next = current.next;  
 System.*out*.println("Value " + value + " deleted from the list.");  
 }  
 }  
  
  
 public void print() {  
 Node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
  
 public static void main(String[] args) {  
 q3 list = new q3();  
 list.insert(1);  
 list.insert(2);  
 list.insert(3);  
 list.insert(4);  
 list.insert(5);  
  
 System.*out*.println();  
 list.print();  
  
 list.delete(3);  
 System.*out*.println();  
 list.print();  
  
 list.delete(6);  
 }  
}

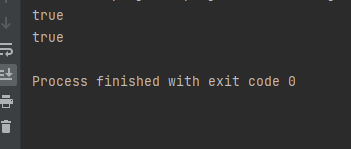


Q4:

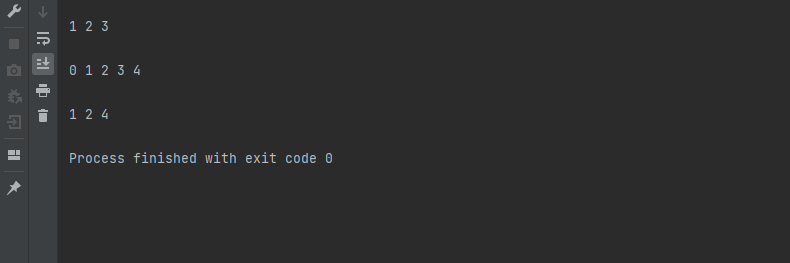
class listNode {  
 int val;  
 listNode next;  
  
 listNode(int val) {  
 this.val = val;  
 }  
}  
  
public class q4 {  
 public static listNode modifyLinkedList(listNode head) {  
 if (head == null || head.next == null) {  
 return head;  
 }  
  
 listNode evenHead = new listNode(0);   
 listNode oddHead = new listNode(0);   
 listNode evenTail = evenHead;  
 listNode oddTail = oddHead;  
  
 listNode current = head;  
  
 while (current != null) {  
 if (current.val % 2 == 0) {  
 evenTail.next = current;  
 evenTail = evenTail.next;  
 } else {  
 oddTail.next = current;  
 oddTail = oddTail.next;  
 }  
 current = current.next;  
 }  
  
   
 evenTail.next = oddHead.next;  
 oddTail.next = null;  
  
 return evenHead.next;  
 }  
  
 public static void printLinkedList(listNode head) {  
 listNode current = head;  
 while (current != null) {  
 System.*out*.print(current.val + " -> ");  
 current = current.next;  
 }  
 System.*out*.println("NULL");  
 }  
  
 public static void main(String[] args) {  
 listNode head = new listNode(17);  
 head.next = new listNode(15);  
 head.next.next = new listNode(8);  
 head.next.next.next = new listNode(12);  
 head.next.next.next.next = new listNode(10);  
 head.next.next.next.next.next = new listNode(5);  
 head.next.next.next.next.next.next = new listNode(4);  
 head.next.next.next.next.next.next.next = new listNode(1);  
 head.next.next.next.next.next.next.next.next = new listNode(7);  
 head.next.next.next.next.next.next.next.next.next = new listNode(6);  
  
 System.*out*.println();  
 *printLinkedList*(head);  
  
 listNode modifiedHead = *modifyLinkedList*(head);  
  
 System.*out*.println();  
 *printLinkedList*(modifiedHead);  
 }  
}  
  


Q5:

class node {  
 int val;  
 node next;  
  
 node(int val) {  
 this.val = val;  
 }  
}  
  
public class q5 {  
  
 public static boolean Palindrome(node head) {  
 if (head == null || head.next == null) {  
 return true;  
 }  
  
  
 int length = 0;  
 node current = head;  
 while (current != null) {  
 length++;  
 current = current.next;  
 }  
  
  
 int mid = length / 2;  
 node firstHalf = head;  
 node secondHalf = head;  
  
 for (int i = 0; i < mid; i++) {  
 secondHalf = secondHalf.next;  
 }  
  
  
 if (length % 2 != 0) {  
 secondHalf = secondHalf.next;  
 }  
  
  
 secondHalf = *reverse*(secondHalf);  
  
  
 while (secondHalf != null) {  
 if (firstHalf.val != secondHalf.val) {  
 return false;  
 }  
 firstHalf = firstHalf.next;  
 secondHalf = secondHalf.next;  
 }  
  
 return true;  
 }  
  
 private static node reverse(node head) {  
 node prev = null;  
 node current = head;  
 while (current != null) {  
 node nextTemp = current.next;  
 current.next = prev;  
 prev = current;  
 current = nextTemp;  
 }  
 return prev;  
 }  
  
 public static void main(String[] args) {  
  
 node list1 = new node(1);  
 list1.next = new node(0);  
 list1.next.next = new node(2);  
 list1.next.next.next = new node(0);  
 list1.next.next.next.next = new node(1);  
  
 System.*out*.println(*Palindrome*(list1));  
  
   
 node list2 = new node('B');  
 list2.next = new node('O');  
 list2.next.next = new node('R');  
 list2.next.next.next = new node('R');  
 list2.next.next.next.next = new node('O');  
 list2.next.next.next.next.next = new node('W');  
 list2.next.next.next.next.next.next = new node('O');  
 list2.next.next.next.next.next.next.next = new node('R');  
 list2.next.next.next.next.next.next.next.next = new node('R');  
 list2.next.next.next.next.next.next.next.next.next = new node('O');  
 list2.next.next.next.next.next.next.next.next.next.next = new node('B');  
  
 System.*out*.println(*Palindrome*(list2));  
 }  
}

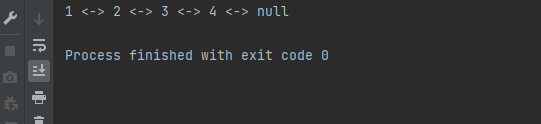


Q6:  
  
class q6 {  
  
  
 class Node {  
 int data;  
 Node next;  
  
 public Node(int data) {  
 this.data = data;  
 this.next = null;  
 }  
 }  
 Node head;  
  
  
 public void AtEnd(int data) {  
 Node newNode = new Node(data);  
 if (head == null) {  
 head = newNode;  
 newNode.next = head;  
 } else {  
 Node current = head;  
 while (current.next != head) {  
 current = current.next;  
 }  
 current.next = newNode;  
 newNode.next = head;  
 }  
 }  
  
  
 public void AtBeginning(int data) {  
 Node newNode = new Node(data);  
 if (head == null) {  
 head = newNode;  
 newNode.next = head;  
 } else {  
 Node current = head;  
 while (current.next != head) {  
 current = current.next;  
 }  
 newNode.next = head;  
 head = newNode;  
 current.next = head;  
 }  
 }  
  
  
 public void AtPosition(int data, int position) {  
 Node newNode = new Node(data);  
 if (position <= 0) {  
 System.*out*.println("Invalid position");  
 return;  
 }  
 if (position == 1) {  
 AtBeginning(data);  
 return;  
 }  
 Node current = head;  
 int count = 1;  
 while (count < position - 1 && current.next != head) {  
 current = current.next;  
 count++;  
 }  
 if (count < position - 1) {  
 System.*out*.println("Position is out of range");  
 return;  
 }  
 newNode.next = current.next;  
 current.next = newNode;  
 }  
  
  
 public void deleteNode(int data) {  
 if (head == null) {  
 System.*out*.println("List is empty");  
 return;  
 }  
 if (head.data == data) {  
 Node current = head;  
 while (current.next != head) {  
 current = current.next;  
 }  
 if (head == head.next) {  
 head = null;  
 } else {  
 head = head.next;  
 current.next = head;  
 }  
 return;  
 }  
 Node current = head;  
 Node prev = null;  
 while (current.next != head) {  
 if (current.data == data) {  
 prev.next = current.next;  
 return;  
 }  
 prev = current;  
 current = current.next;  
 }  
 if (current.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 current = head;  
 do {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 } while (current != head);  
 System.*out*.println();  
 }  
  
 public static void main(String[] args) {  
 q6 list = new q6();  
  
 list.AtEnd(1);  
 list.AtEnd(2);  
 list.AtEnd(3);  
  
 System.*out*.println();  
 list.print();  
  
 list.AtBeginning(0);  
 list.AtPosition(4, 5);  
  
 System.*out*.println();  
 list.print();  
  
 list.deleteNode(0);  
 list.deleteNode(3);  
  
 System.*out*.println();  
 list.print();  
 }  
  
}



Q7:

class node {  
 int data;  
 node next;  
 node prev;  
  
 public node(int data) {  
 this.data = data;  
 this.next = null;  
 this.prev = null;  
 }  
}  
  
class q7{  
  
 node head;  
 node tail;  
  
  
 public q7() {  
 this.head = null;  
 this.tail = null;  
 }  
  
  
 public static q7 concatenate(q7 L, q7 M) {  
  
 if (L == null || L.head == null) {  
 return M;  
 }  
 if (M == null || M.head == null) {  
 return L;  
 }  
  
  
 L.tail.next = M.head;  
 M.head.prev = L.tail;  
  
  
 L.tail = M.tail;  
 M.head = null;  
  
 return L;  
 }  
  
  
 public void print() {  
 node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " <-> ");  
 current = current.next;  
 }  
 System.*out*.println("null");  
 }  
  
 public static void main(String[] args) {  
 q7 L = new q7();  
 q7 M = new q7();  
  
  
 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;  
  
  
 q7 List = q7.*concatenate*(L, M);  
  
  
 List.print();  
 }  
}



Q8:

class Node {  
 int data;  
 Node next;  
  
 public Node(int data) {  
 this.data = data;  
 this.next = null;  
 }  
}  
  
public class q8 {  
  
 public static void modify(Node head) {  
 if (head == null || head.next == null) {  
 return;  
 }  
  
 Node current = head;  
 Node alternate = head.next;  
 Node newHead = null;  
  
 while (current != null && alternate != null) {  
 current.next = alternate.next;  
 alternate.next = newHead;  
 newHead = alternate;  
 alternate = current.next;  
  
 if (alternate != null) {  
 current = alternate;  
 alternate = current.next;  
 }  
 }  
  
  
 current.next = newHead;  
  
  
 Node temp = head;  
 while (temp != null) {  
 System.*out*.print(temp.data + " ");  
 temp = temp.next;  
 }  
 }  
  
 public static void main(String[] args) {  
  
 Node head = new Node(10);  
 head.next = new Node(4);  
 head.next.next = new Node(9);  
 head.next.next.next = new Node(1);  
 head.next.next.next.next = new Node(3);  
 head.next.next.next.next.next = new Node(5);  
 head.next.next.next.next.next.next = new Node(9);  
 head.next.next.next.next.next.next.next = new Node(4);  
  
  
 Node temp = head;  
 while (temp != null) {  
 System.*out*.print(temp.data + " ");  
 temp = temp.next;  
 }  
 System.*out*.println();  
  
 *modify*(head);  
 }  
}

