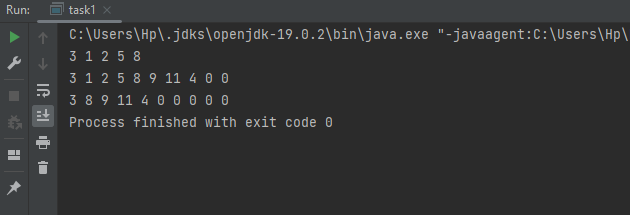
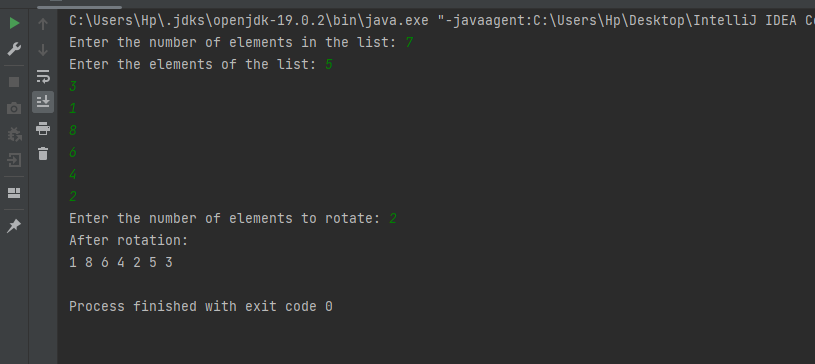
**LAB 4**  
**LAIBA FATIMA 22k-5195**  
  
  
Task 1:  
public class task1 {  
 public static int[] growSize(int [] array)  
 {  
 int sizeofarray = array.length;  
  
 int temp[] = null;  
  
 temp = new int[array.length \* 2];  
 {  
 for (int i = 0; i < array.length; i++)  
 {  
  
 temp[i] = array[i];  
 }  
 }  
  
 array = temp;  
 sizeofarray= array.length \* 2;  
  
 return array;  
 }  
  
 public static void printArray(int[] array){  
 int n=array.length;  
 for( int i= 0; i<n; i++){  
 System.*out*.print(array[i] + " ");  
 }  
 }  
  
  
 public static void main(String[] args) {  
 int [] array = {3,1,2,5,8};  
 *printArray*(array);  
  
 array = *growSize*(array);  
  
 array[4+1] = 9;  
 array[4+2] = 11;  
 array[4+3] = 4;  
  
 System.*out*.println();  
 *printArray*(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();  
 *printArray*(array);  
 }  
}



Task 2:

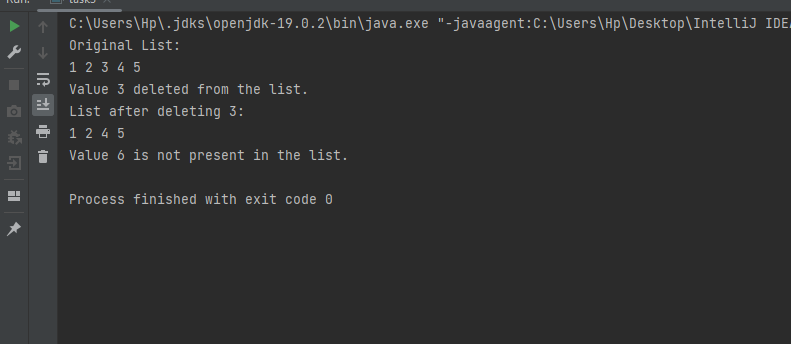
import java.util.Scanner;  
  
class Node {

int data;  
 Node next;  
  
 Node(int data) {  
 this.data = data;  
 next = null;  
 }  
}  
public class task2 {  
 public static Node rotateLinkedList(Node head, int n) {  
 if (head == null || n == 0) {  
 return head;  
 }  
  
 Node current = head;  
 int count = 1;  
  
 while (count < n && current != null) {  
 current = current.next;  
 count++;  
 }  
  
 if (current == null) {  
 return head;  
 }  
  
 Node nthNode = current;  
 while (current.next != null) {  
 current = current.next;  
 }  
  
 current.next = head;  
 head = nthNode.next;  
 nthNode.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 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 elements to rotate: ");  
 int k = scanner.nextInt();  
  
 head = *rotateLinkedList*(head, k);  
  
 System.*out*.println("After rotation:");  
 *printList*(head);  
  
 }  
}



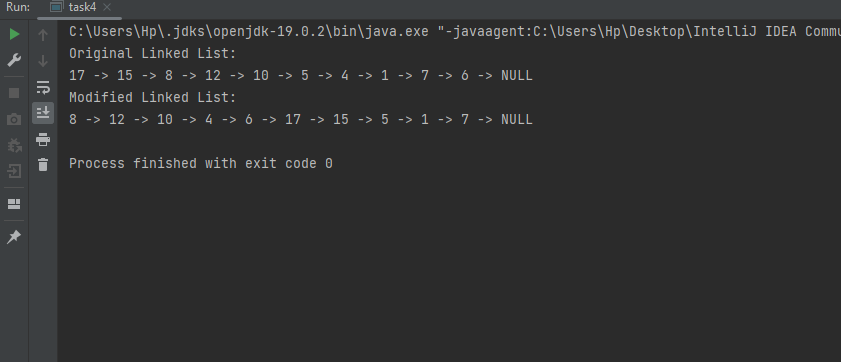
Task 3:

public class task3 {  
 class Node {  
 int data;  
 Node next;  
  
 public Node(int data) {  
 this.data = data;  
 this.next = null;  
 }  
 }  
  
 Node head;  
  
 public task3() {  
 this.head = null;  
 }  
  
  
 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 display() {  
 Node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
  
 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 class Main {  
 public static void main(String[] args) {  
 task3 list = new task3();  
 list.insert(1);  
 list.insert(2);  
 list.insert(3);  
 list.insert(4);  
 list.insert(5);  
  
 System.*out*.println("Original List:");  
 list.display();  
  
 list.delete(3);  
 System.*out*.println("List after deleting 3:");  
 list.display();  
  
 list.delete(6);  
 }  
 }  
  
  
  
  
  
  
}



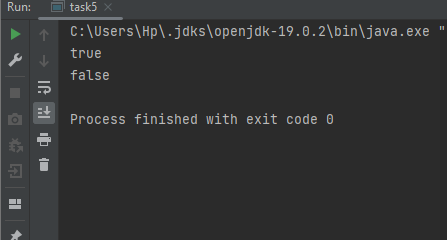
Task 4:

class listNode {  
 int val;  
 listNode next;  
  
 listNode(int val) {  
 this.val = val;  
 }  
}  
  
  
public class task4 {  
  
 static class LinkedList {  
 listNode head;  
 public void modifyLinkedList() {  
  
  
 if (head == null || head.next == null) {  
 return ;  
 }  
  
 listNode evenHead = null;  
 listNode oddHead = null;  
 listNode evenTail = null;  
 listNode oddTail = null;  
  
 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;  
 }  
  
 if (evenHead != null) {  
 evenTail.next = oddHead;  
 head = evenHead;  
 }  
  
  
  
 }  
  
 public void insertAtLast(int data) {  
 listNode newNode = new listNode(data);  
 if (head == null) {  
 head = newNode;  
 } else {  
 listNode curr = head;  
 while (curr.next != null) {  
 curr = curr.next;  
 }  
 curr.next = newNode;  
 }  
 }  
  
 public void printLinkedList() {  
 listNode current = head;  
 while (current != null) {  
 System.*out*.print(current.val + " -> ");  
 current = current.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*.println("Original Linked List:");  
 list.printLinkedList();  
  
 list.modifyLinkedList();  
 System.*out*.println("Modified Linked List:");  
 list.printLinkedList();  
  
 }  
  
   
}



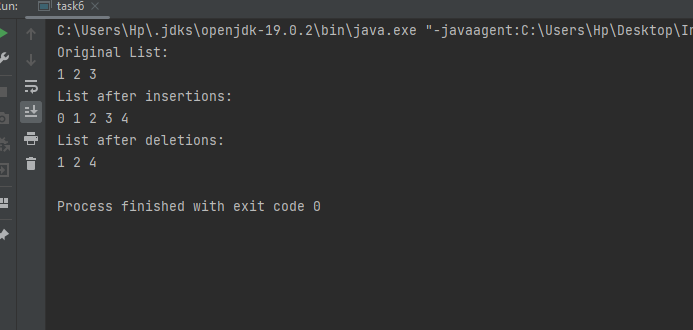
Task 5:

class listNode {  
 int val;  
 listNode next;  
  
 listNode(int val) {  
 this.val = val;  
 }  
}  
  
public class task5 {  
  
 static class LinkedList {  
 listNode head;  
  
  
  
 public void insert(int data) {  
 listNode newNode = new listNode((char) data);  
 if (head == null) {  
 head = newNode;  
 } else {  
 listNode curr = head;  
 while (curr.next != null) {  
 curr = curr.next;  
 }  
 curr.next = newNode;  
 }  
 }  
  
 public boolean Palindrome() {  
 if (head == null || head.next == null) {  
 return true;  
 }  
  
  
 listNode slow = head;  
 listNode fast = head;  
  
  
 while (fast != null && fast.next != null) {  
 slow = slow.next;  
 fast = fast.next.next;  
 }  
 listNode secondhalf = reverselist(slow);  
 listNode firsthalf = head;  
  
  
 while (secondhalf != null) {  
 if (firsthalf.val != secondhalf.val) {  
 return false;  
 }  
 firsthalf = firsthalf.next;  
 secondhalf = secondhalf.next;  
 }  
  
  
 return true;  
 }  
  
 listNode reverselist(listNode head) {  
 listNode prev = null;  
 listNode curr = head;  
 while (curr != null) {  
 listNode nextTemp = curr.next;  
 curr.next = prev;  
 prev = curr;  
 curr = nextTemp;  
 }  
 return prev;  
 }  
  
  
  
  
 }  
  
  
  
  
 public static void main(String[] args) {  
 LinkedList list1 = new LinkedList();  
 list1.insert('B');  
 list1.insert('O');  
 list1.insert('R');  
 list1.insert('R');  
 list1.insert('O');  
 list1.insert('W');  
 list1.insert('O');  
 list1.insert('R');  
 list1.insert('R');  
 list1.insert('O');  
 list1.insert('B');  
  
 boolean answer = list1.Palindrome();  
 System.*out*.println(answer);  
  
 LinkedList list2 = new LinkedList();  
 list2.insert('A');  
 list2.insert('B');  
 list2.insert('C');  
  
 answer= list2.Palindrome();  
 System.*out*.println(answer);  
 }  
}



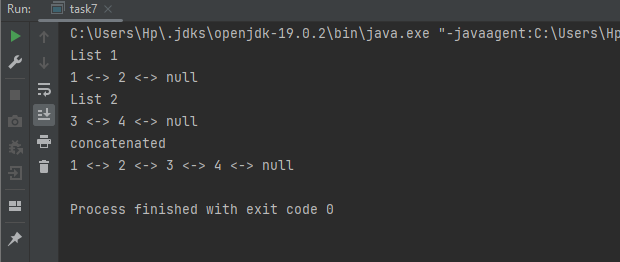
Task 6:

class task6 {  
  
 class listNode {  
 int data;  
 listNode next;  
  
 public listNode(int data) {  
 this.data = data;  
 this.next = null;  
 }  
 }  
 listNode head;  
  
  
 public void insertAtEnd(int data) {  
 listNode newListNode = new listNode(data);  
 if (head == null) {  
 head = newListNode;  
 newListNode.next = head;  
 } else {  
 listNode current = head;  
 while (current.next != head) {  
 current = current.next;  
 }  
 current.next = newListNode;  
 newListNode.next = head;  
 }  
 }  
  
  
 public void insertAtBeginning(int data) {  
 listNode newListNode = new listNode(data);  
 if (head == null) {  
 head = newListNode;  
 newListNode.next = head;  
 } else {  
 listNode current = head;  
 while (current.next != head) {  
 current = current.next;  
 }  
 newListNode.next = head;  
 head = newListNode;  
 current.next = head;  
 }  
 }  
  
  
 public void insertAtPosition(int data, int position) {  
 listNode newListNode = new listNode(data);  
 if (position <= 0) {  
 System.*out*.println("Invalid position");  
 return;  
 }  
 if (position == 1) {  
 insertAtBeginning(data);  
 return;  
 }  
 listNode 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;  
 }  
 newListNode.next = current.next;  
 current.next = newListNode;  
 }  
  
  
 public void deleteNode(int data) {  
 if (head == null) {  
 System.*out*.println("List is empty");  
 return;  
 }  
 if (head.data == data) {  
 listNode current = head;  
 while (current.next != head) {  
 current = current.next;  
 }  
 if (head == head.next) {  
 head = null;  
 } else {  
 head = head.next;  
 current.next = head;  
 }  
 return;  
 }  
 listNode current = head;  
 listNode 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 printList() {  
 if (head == null) {  
 System.*out*.println("List is empty");  
 return;  
 }  
 listNode current = head;  
 do {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 } while (current != head);  
 System.*out*.println();  
 }  
  
 public static void main(String[] args) {  
 task6 list = new task6();  
  
 list.insertAtEnd(1);  
 list.insertAtEnd(2);  
 list.insertAtEnd(3);  
  
 System.*out*.println("Original List:");  
 list.printList();  
  
 list.insertAtBeginning(0);  
 list.insertAtPosition(4, 5);  
  
 System.*out*.println("List after insertions:");  
 list.printList();  
  
 list.deleteNode(0);  
 list.deleteNode(3);  
  
 System.*out*.println("List after deletions:");  
 list.printList();  
 }  
  
}



Task 7:

class node {  
 int data;  
 node next;  
 node prev;  
  
 public node(int data) {  
 this.data = data;  
 this.next = null;  
 this.prev = null;  
 }  
}  
  
public class task7 {  
  
 node head;  
 node tail;  
  
  
 public task7() {  
 this.head = null;  
 this.tail = null;  
 }  
  
  
 public static task7 concatenate(task7 L, task7 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 display() {  
 node current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " <-> ");  
 current = current.next;  
 }  
 System.*out*.println("null");  
 }  
  
 public static void main(String[] args) {  
 task7 L = new task7();  
 task7 M = new task7();  
  
  
 L.head = new node(1);  
 L.head.next = new node(2);  
 L.tail = L.head.next;  
  
 System.*out*.println("List 1");  
 L.display();  
  
 M.head = new node(3);  
 M.head.next = new node(4);  
 M.tail = M.head.next;  
  
 System.*out*.println("List 2");  
 M.display();  
  
 task7 concatenatedList = task7.*concatenate*(L, M);  
 System.*out*.println("concatenated");  
  
 concatenatedList.display();  
 }  
}



Task 8:

public class task8 {  
  
 static class listNode {  
 int data;  
 listNode next;  
  
 public listNode(int data) {  
 this.data = data;  
 this.next = null;  
 }  
  
 }  
 static class LinkedList {  
 listNode head;  
  
 public void printList() {  
 listNode current = head;  
 while (current != null) {  
 System.*out*.print(current.data + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
 public void insert(int data) {  
 listNode newNode = new listNode(data);  
 if (head == null) {  
 head = newNode;  
 } else {  
 listNode curr = head;  
 while (curr.next != null) {  
 curr = curr.next;  
 }  
 curr.next = newNode;  
 }  
 }  
 public void modify() {  
 if (head == null || head.next == null) {  
 return;  
 }  
  
 listNode current = head;  
 listNode alternate = head.next;  
 listNode 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;  
  
 }  
 }

public static void main(String[] args) {  
 LinkedList list = new LinkedList();  
 list.insert(10);  
 list.insert(4);  
 list.insert(9);  
 list.insert(1);  
 list.insert(3);  
 list.insert(5);  
 list.insert(9);  
 list.insert(4);  
  
 System.*out*.println("Original Linked List:");  
 list.printList();

list.modify();  
  
 System.*out*.println("\nModified Linked List:");  
 list.printList();  
 }

}

