**// DOUBLY LINKED LIST TASK //**

package dsa\_2025;

public class DoublyNode {

int data;

DoublyNode next;

DoublyNode prev;

public DoublyNode(int data) {

this.data = data;

this.next = null;

this.prev = null;

System.out.println("Node created Successfully");

}

public void display(DoublyNode head) {

DoublyNode temp = head;

while (temp != null) {

System.out.print(" <-> " + temp.data);

temp = temp.next;

}

System.out.println();

}

public DoublyNode addNodeAtStart(DoublyNode head, int data) {

DoublyNode newNode = new DoublyNode(data);

if (head != null) {

newNode.next = head;

head.prev = newNode;

}

return newNode;

}

public void addNodeAtEnd(DoublyNode head, int data) {

DoublyNode newNode = new DoublyNode(data);

DoublyNode temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNode;

newNode.prev = temp;

}

public void addNodeAtPos(DoublyNode head, int data, int pos) {

DoublyNode newNode = new DoublyNode(data);

DoublyNode temp = head;

for (int i = 1; i < pos - 1 && temp.next != null; i++) {

temp = temp.next;

}

newNode.next = temp.next;

if (temp.next != null) {

temp.next.prev = newNode;

}

temp.next = newNode;

newNode.prev = temp;

}

public DoublyNode deleteFirstNode(DoublyNode head) {

if (head == null) return null;

head = head.next;

if (head != null) {

head.prev = null;

}

return head;

}

public void deleteLastNode(DoublyNode head) {

if (head == null || head.next == null) return;

DoublyNode temp = head;

while (temp.next.next != null) {

temp = temp.next;

}

temp.next = null;

}

public void deleteNodeAtPos(DoublyNode head, int pos) {

if (head == null) return;

DoublyNode temp = head;

for (int i = 1; i < pos - 1 && temp.next != null; i++) {

temp = temp.next;

}

if (temp.next == null) return;

DoublyNode nodeToDelete = temp.next;

temp.next = nodeToDelete.next;

if (nodeToDelete.next != null) {

nodeToDelete.next.prev = temp;

}

}

public static void main(String args[]) {

DoublyNode first = new DoublyNode(10);

DoublyNode second = new DoublyNode(20);

DoublyNode third = new DoublyNode(30);

DoublyNode fourth = new DoublyNode(40);

DoublyNode head = first;

first.next = second;

second.prev = first;

second.next = third;

third.prev = second;

third.next = fourth;

fourth.prev = third;

System.out.println("\n# Print Data of Doubly Linked List:");

head.display(head);

head = head.addNodeAtStart(head, 5);

System.out.println("\n## After inserting at the beginning:");

head.display(head);

head.addNodeAtEnd(head, 50);

System.out.println("\n### After inserting at the end:");

head.display(head);

head.addNodeAtPos(head, 25, 3);

System.out.println("\n#### After inserting at a specific position:");

head.display(head);

head = head.deleteFirstNode(head);

System.out.println("\n##### After deleting the first node:");

head.display(head);

head.deleteLastNode(head);

System.out.println("\n###### After deleting the last node:");

head.display(head);

head.deleteNodeAtPos(head, 3);

System.out.println("\n####### After deleting a specific node:");

head.display(head);

}

}

// OUTPUT

