Assignment 13 Solution - Linked List| DSA

Question 1

Given two linked list of the same size, the task is to create a new linked list using those linked lists. The condition is that the greater node among both linked list will be added to the new linked list.

Examples:
Input: list1 = 5->2->3->8
list2 = 1->7->4->5
Output: New list = 5->7->4->8

Input:list1 = 2->8->9->3
list2 = 5->3->6->4
Output: New list = 5->8->9->4

Solution Code:

```
package in.ineuron.pptAssignment13;
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
        next = null;
    }
}
class LinkedList {
    Node head;

    void add(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;
        }
    }

    static LinkedList createNewList(LinkedList list1, LinkedList list2) {
        LinkedList newList = new LinkedList();
        Node current1 = list1.head;
        Node current2 = list2.head;
```

```
                while (current1 != null && current2 != null) {
                        int data1 = current1.data;
                        int data2 = current2.data;

                        if (data1 >= data2) {
                                newList.add(data1);
                        } else {
                                newList.add(data2);
                        }

                        current1 = current1.next;
                        current2 = current2.next;
                }

                return newList;
        }

        void display() {
                Node current = head;
                while (current != null) {
                        System.out.print(current.data + " ");
                        current = current.next;
                }
                System.out.println();
        }
}
class CreateNewList_1 {
        public static void main(String[] args) {
                LinkedList list1 = new LinkedList();
                list1.add(5);
                list1.add(2);
                list1.add(3);
                list1.add(8);

                LinkedList list2 = new LinkedList();
                list2.add(1);
                list2.add(7);
                list2.add(4);
                list2.add(5);

                LinkedList newList = LinkedList.createNewList(list1, list2);

                System.out.print("New list = ");
                newList.display();
        }
}
```

Question 2

Write a function that takes a list sorted in non-decreasing order and deletes any duplicate nodes from the list. The list should only be traversed once.

For example if the linked list is 11->11->11->21->43->43->60 then removeDuplicates() should convert the list to 11->21->43->60.

Example 1:

Input:

LinkedList: 11->11->11->21->43->43->60

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

Example 2:

Input:

LinkedList: 10->12->12->25->25->25->34

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

Solution Code:

```
package in.ineuron.pptAssignment13;

class Node2 {
    int data;
    Node2 next;

    Node2(int data) {
        this.data = data;
        next = null;
    }
}

class LinkedList2 {
    Node2 head;

    void add(int data) {
        Node2 newNode = new Node2(data);
        if (head == null) {
            head = newNode;
        } else {
            Node2 current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }
```

```
        void removeDuplicates() {
            if (head == null || head.next == null) {
                return;
            }

            Node2 current = head;
            while (current != null && current.next != null) {
                if (current.data == current.next.data) {
                    current.next = current.next.next;
                } else {
                    current = current.next;
                }
            }
        }

        void display() {
            Node2 current = head;
            while (current != null) {
                System.out.print(current.data + " ");
                current = current.next;
            }
            System.out.println();
        }
    }

    public class RemoveDuplicates_2 {
        public static void main(String[] args) {
            LinkedList2 list = new LinkedList2();
            list.add(11);
            list.add(11);
            list.add(11);
            list.add(21);
            list.add(43);
            list.add(43);
            list.add(60);

            System.out.print("Input: ");
            list.display();

            list.removeDuplicates();

            System.out.print("Output: ");
            list.display();
        }
    }
```

Question 3

Given a linked list of size N. The task is to reverse every k nodes (where k is an input to the function) in the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should be considered as a group and must be reversed (See Example 2 for clarification).

Example 1:
Input:
LinkedList: 1->2->2->4->5->6->7->8
K = 4
Output:4 2 2 1 8 7 6 5

Explanation:
The first 4 elements 1,2,2,4 are reversed first and then the next 4 elements 5,6,7,8. Hence, the resultant linked list is 4->2->2->1->8->7->6->5.

Example 2:
Input:
LinkedList: 1->2->3->4->5
K = 3
Output:3 2 1 5 4
Explanation:
The first 3 elements are 1,2,3 are reversed first and then elements 4,5 are reversed. Hence, the resultant linked list is 3->2->1->5->4.

Solution Code:

```
package in.ineuron.pptAssignment13;

class Node3 {
	int data;
	Node3 next;

	Node3(int data) {
		this.data = data;
		next = null;
	}
}

class LinkedList3 {
	Node3 head;

	void add(int data) {
		Node3 newNode = new Node3(data);
		if (head == null) {
			head = newNode;
		} else {
```

```
                        Node3 current = head;
                        while (current.next != null) {
                                current = current.next;
                        }
                        current.next = newNode;
                }
        }

        Node3 reverseKNodes(Node3 head, int k) {
                Node3 current = head;
                Node3 next = null;
                Node3 prev = null;
                int count = 0;

                while (count < k && current != null) {
                        next = current.next;
                        current.next = prev;
                        prev = current;
                        current = next;
                        count++;
                }

                if (next != null) {
                        head.next = reverseKNodes(next, k);
                }

                return prev;
        }

        void display() {
                Node3 current = head;
                while (current != null) {
                        System.out.print(current.data + " ");
                        current = current.next;
                }
                System.out.println();
        }
}

public class ReverseKNodes_3 {
        public static void main(String[] args) {
                LinkedList3 list = new LinkedList3();
                list.add(1);
                list.add(2);
                list.add(2);
                list.add(4);
                list.add(5);
                list.add(6);
                list.add(7);
                list.add(8);

                int k = 4;

                System.out.print("Input: ");
                list.display();

                list.head = list.reverseKNodes(list.head, k);

                System.out.print("Output: ");
                list.display();
        }
}
```

Question 4

Given a linked list, write a function to reverse every alternate k nodes (where k is an input to the function) in an efficient way. Give the complexity of your algorithm.

Example:

Inputs: 1->2->3->4->5->6->7->8->9->NULL and k = 3

Output: 3->2->1->4->5->6->9->8->7->NULL.

Solution Code:

```
package in.ineuron.pptAssignment13;

class Node4 {
    int data;
    Node4 next;

    Node4(int data) {
        this.data = data;
        next = null;
    }
}

class LinkedList4 {
    Node4 head;

    void add(int data) {
        Node4 newNode = new Node4(data);
        if (head == null) {
            head = newNode;
        } else {
            Node4 current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    Node4 reverseAlternateKNodes(Node4 head, int k) {
        if (head == null || head.next == null || k <= 1) {
            return head;
        }

        Node4 current = head;
        Node4 prev = null;
        Node4 next = null;
        int count = 0;

        // Reverse k nodes
        while (current != null && count < k) {
            next = current.next;
            current.next = prev;
            prev = current;
            current = next;
            count++;
        }

        // Connect the reversed k nodes to the next set of k nodes
        if (head != null) {
            head.next = current;
        }

        // Skip the next k nodes
        count = 0;
        while (count < k - 1 && current != null) {
            current = current.next;
            count++;
        }

        // Recursively reverse the next set of alternate k nodes
        if (current != null) {
            current.next = reverseAlternateKNodes(current.next, k);
        }

        return prev;
    }
```

```
        void display() {
            Node4 current = head;
            while (current != null) {
                System.out.print(current.data + " ");
                current = current.next;
            }
            System.out.println();
        }
    }

    public class ReverseAlternateKNodes_4 {
        public static void main(String[] args) {
            LinkedList4 list = new LinkedList4();
            list.add(1);
            list.add(2);
            list.add(3);
            list.add(4);
            list.add(5);
            list.add(6);
            list.add(7);
            list.add(8);
            list.add(9);

            int k = 3;

            System.out.print("Input: ");
            list.display();

            list.head = list.reverseAlternateKNodes(list.head, k);

            System.out.print("Output: ");
            list.display();
        }
    }
```

Question 5

Given a linked list and a key to be deleted. Delete last occurrence of key from linked. The list may have duplicates.

Examples:
Input: 1->2->3->5->2->10, key = 2
Output: 1->2->3->5->10

Solution Code:

```
package in.ineuron.pptAssignment13;
class Node5 {
    int data;
    Node5 next;

    Node5(int data) {
        this.data = data;
        next = null;
    }
}
class LinkedList5 {
    Node5 head;

    void add(int data) {
        Node5 newNode = new Node5(data);
        if (head == null) {
            head = newNode;
        } else {
            Node5 current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    void deleteLastOccurrence(int key) {
        if (head == null) {
            return;
        }

        Node5 lastOccurrence = null;
        Node5 current = head;
        Node5 prev = null;
        Node5 lastPrev = null;

        while (current != null) {
            if (current.data == key) {
                lastOccurrence = current;
                lastPrev = prev;
            }
            prev = current;
            current = current.next;
        }

        if (lastOccurrence != null) {
            if (lastPrev != null) {
                lastPrev.next = lastOccurrence.next;
            } else {
                head = lastOccurrence.next;
            }
        }
    }

    void display() {
        Node5 current = head;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println();
    }
}
public class DeleteLastOccurrence_5 {
    public static void main(String[] args) {
        LinkedList5 list = new LinkedList5();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(5);
        list.add(2);
        list.add(10);

        int key = 2;

        System.out.print("Input: ");
        list.display();

        list.deleteLastOccurrence(key);

        System.out.print("Output: ");
        list.display();
    }
}
```

Question 6

Given two sorted linked lists consisting of N and M nodes respectively. The task is to merge both of the lists (in place) and return the head of the merged list.

Examples:
Input: a: 5->10->15, b: 2->3->20
Output: 2->3->5->10->15->20

Input: a: 1->1, b: 2->4
Output: 1->1->2->4

Solution Code:

```
package in.ineuron.pptAssignment13;

class Node6 {
	int data;
	Node6 next;

	Node6(int data) {
		this.data = data;
		next = null;
	}
}

class LinkedList6 {
	Node6 head;

	void add(int data) {
		Node6 newNode = new Node6(data);
		if (head == null) {
			head = newNode;
		} else {
			Node6 current = head;
			while (current.next != null) {
				current = current.next;
			}
			current.next = newNode;
		}
	}

	Node6 mergeSortedLists(Node6 a, Node6 b) {
		if (a == null) {
			return b;
		}
		if (b == null) {
			return a;
```

```
                }

                Node6 result;
                if (a.data <= b.data) {
                        result = a;
                        result.next = mergeSortedLists(a.next, b);
                } else {
                        result = b;
                        result.next = mergeSortedLists(a, b.next);
                }
                return result;
        }

        void display() {
                Node6 current = head;
                while (current != null) {
                        System.out.print(current.data + " ");
                        current = current.next;
                }
                System.out.println();
        }
}

public class MergeSortedLists_6 {
        public static void main(String[] args) {
                LinkedList6 list1 = new LinkedList6();
                list1.add(5);
                list1.add(10);
                list1.add(15);

                LinkedList6 list2 = new LinkedList6();
                list2.add(2);
                list2.add(3);
                list2.add(20);

                System.out.print("Input 1: ");
                list1.display();
                System.out.print("Input 2: ");
                list2.display();

                LinkedList6 mergedList = new LinkedList6();
                mergedList.head = mergedList.mergeSortedLists(list1.head, list2.head);

                System.out.print("Merged List: ");
                mergedList.display();
        }
}
```

Question 7

Given a Doubly Linked List, the task is to reverse the given Doubly Linked List.

Example:
Original Linked list 10 8 4 2
Reversed Linked list 2 4 8 10

Solution Code:

```
package in.ineuron.pptAssignment13;

class Node7 {
    int data;
    Node7 prev;
    Node7 next;

    Node7(int data) {
        this.data = data;
        prev = null;
        next = null;
    }
}

class DoublyLinkedList7 {
    Node7 head;

    void add(int data) {
        Node7 newNode = new Node7(data);
        if (head == null) {
            head = newNode;
        } else {
            Node7 current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
            newNode.prev = current;
        }
    }

    void reverse() {
        Node7 current = head;
        Node7 temp = null;

        while (current != null) {
            temp = current.prev;
            current.prev = current.next;
            current.next = temp;
            current = current.prev;
        }

        if (temp != null) {
            head = temp.prev;
        }
    }

    void display() {
        Node7 current = head;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println();
    }
}

public class ReverseLinkedList_7 {
    public static void main(String[] args) {
        DoublyLinkedList7 list = new DoublyLinkedList7();
        list.add(10);
        list.add(8);
        list.add(4);
        list.add(2);

        System.out.print("Original Linked List: ");
        list.display();

        list.reverse();

        System.out.print("Reversed Linked List: ");
        list.display();
    }
}
```

Question 8

Given a doubly linked list and a position. The task is to delete a node from given position in a doubly linked list.

Example 1:
Input:
LinkedList = 1 <--> 3 <--> 4
x = 3
Output:1 3

Explanation: After deleting the node at position 3 (position starts from 1), the linked list will be now as 1->3.

Example 2:
Input:
LinkedList = 1 <--> 5 <--> 2 <--> 9
x = 1
Output:5 2 9

Solution Code:

```
package in.ineuron.pptAssignment13;

class Node8 {
    int data;
    Node8 prev;
    Node8 next;

    Node8(int data) {
        this.data = data;
        prev = null;
        next = null;
    }
}

class DoublyLinkedList {
    Node8 head;

    void add(int data) {
        Node8 newNode = new Node8(data);
        if (head == null) {
            head = newNode;
        } else {
            Node8 current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
            newNode.prev = current;
        }
    }

    void deleteNode(int position) {
        if (head == null) {
            return;
        }

        Node8 current = head;
        int count = 1;

        // Traverse to the node to be deleted
        while (current != null && count < position) {
            current = current.next;
            count++;
        }

        // If position exceeds the number of nodes
        if (current == null) {
            return;
        }

        // If the node to be deleted is the head node
        if (current == head) {
            head = head.next;
            if (head != null) {
                head.prev = null;
            }
            return;
        }

        // Update the prev and next pointers of adjacent nodes
        if (current.prev != null) {
            current.prev.next = current.next;
        }
        if (current.next != null) {
            current.next.prev = current.prev;
        }
    }

    void display() {
        Node8 current = head;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
```

```
                }
                System.out.println();
        }
}

public class DeleteNode_8 {
        public static void main(String[] args) {
                DoublyLinkedList list = new DoublyLinkedList();
                list.add(1);
                list.add(3);
                list.add(4);

                System.out.print("Input: ");
                list.display();

                int position = 3;
                list.deleteNode(position);

                System.out.print("Output: ");
                list.display();
        }
}
```