Assignment 12 Solution - Linked List| DSA

Question 1

Given a singly linked list, delete middle of the linked list. For example, if given linked list is 1-2-3-4-5 then linked list should be modified to 1-2-4-5.If there are even nodes, then there would be two middle nodes, we need to delete the second middle element. For example, if given linked list is 1-2-3-4-5-6 then it should be modified to 1-2-3-5-6.If the input linked list is NULL or has 1 node, then it should return NULL

Example 1:
Input:
LinkedList: 1-2-3-4-5
Output:1 2 4 5

Example 2:
Input:
LinkedList: 2-4-6-7-5-1
Output:2 4 6 5 1

Solution Code:

```
package in.ineuron.pptAssignment112;
class Node {
	int data;
	Node next;

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

class LinkedList {
	Node head;

	public void deleteMiddle() {
		if (head == null || head.next == null) {
			// Empty list or single node, nothing to delete
			return;
		}

		Node slowPtr = head;
		Node fastPtr = head;
		Node prevPtr = null;

		while (fastPtr != null && fastPtr.next != null) {
			fastPtr = fastPtr.next.next;
			prevPtr = slowPtr;
```

```
                slowPtr = slowPtr.next;
            }

            // Delete the middle node(s)
            prevPtr.next = slowPtr.next;
            slowPtr.next = null;
        }

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

    public class DeletingMiddleElement_1 {
        public static void main(String[] args) {
            LinkedList list = new LinkedList();
            list.head = new Node(1);
            list.head.next = new Node(2);
            list.head.next.next = new Node(3);
            list.head.next.next.next = new Node(4);
            list.head.next.next.next.next = new Node(5);

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

            list.deleteMiddle();

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

Question 2

Given a linked list of N nodes. The task is to check if the linked list has a loop. Linked list can contain self-loop.

Example 1:
Input: N = 3
value[] = {1,3,4}
x(position at which tail is connected) = 2
Output: True
Explanation: In above test case N = 3.The linked list with nodes N = 3 is given. Then value of x=2 is given which means last node is connected with x-th node of linked list. Therefore, there exists a loop.

Example 2:
Input: N = 4
value[] = {1,8,3,4}
x = 0
Output: False
Explanation: For N = 4 ,x = 0 means then last Node-next = NULL, then the Linked list does not contain any loop.

Solution Code:

```
package in.ineuron.pptAssignment112;

class Node02 {
    int data;
    Node02 next;

    public Node02(int data) {
        this.data = data;
        next = null;
    }
}

class LinkedList02 {
    Node02 head;

    public void addNode(int data) {
        Node02 newNode = new Node02(data);
        if (head == null) {
            head = newNode;
        } else {
            Node02 current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
```

```
		}
	}

	public boolean detectLoop() {
		if (head == null || head.next == null) {
			// Empty list or single node, no loop
			return false;
		}

		Node02 slowPtr = head;
		Node02 fastPtr = head;

		while (fastPtr != null && fastPtr.next != null) {
			slowPtr = slowPtr.next;
			fastPtr = fastPtr.next.next;

			if (slowPtr == fastPtr) {
				// Loop detected
				return true;
			}
		}

		// No loop found
		return false;
	}
}

public class CheckLinkListLOOP_2 {
	public static void main(String[] args) {
		LinkedList02 list = new LinkedList02();
		list.addNode(1);
		list.addNode(3);
		list.addNode(4);

		// Creating a loop by connecting the tail to the second node
		list.head.next.next.next = list.head.next;

		boolean hasLoop = list.detectLoop();
		System.out.println("Does the linked list have a loop? " + hasLoop);
	}
}
```

Question 3

Given a linked list consisting of L nodes and given a number N. The task is to find the Nth node from the end of the linked list.

Example 1:

Input: N = 2
LinkedList: 1-2-3-4-5-6-7-8-9

Output:8

Explanation: In the first example, there are 9 nodes in linked list and we need to find 2nd node from end. 2nd node from end is 8.

Example 2:

Input: N = 5
LinkedList: 10-5-100-5

Output:-1

Explanation: In the second example, there are 4 nodes in the linked list and we need to find 5th from the end. Since 'n' is more than the number of nodes in the linked list, the output is -1.

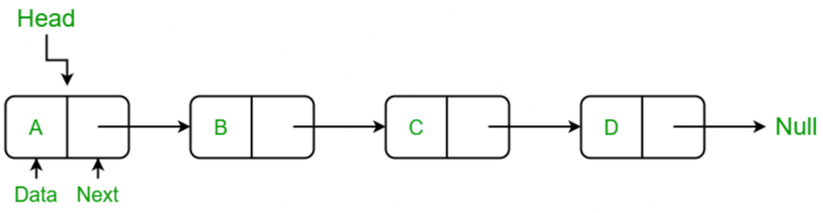
Solution Code:

```
package in.ineuron.pptAssignment112;

class Node03 {
    int data;
    Node03 next;

    public Node03(int data) {
        this.data = data;
        next = null;
    }
}

class LinkedList03 {
    Node03 head;

    public void addNode(int data) {
        Node03 newNode = new Node03(data);
        if (head == null) {
            head = newNode;
        } else {
            Node03 current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    public int findNthFromEnd(int n) {
```

```
		if (head == null) {
			// Empty list
			return -1;
		}

		Node03 slowPtr = head;
		Node03 fastPtr = head;

		// Move the fast pointer n positions ahead
		for (int i = 0; i < n; i++) {
			if (fastPtr == null) {
				// n is greater than the number of nodes
				return -1;
			}
			fastPtr = fastPtr.next;
		}

		// Move both pointers until the fast pointer reaches the end
		while (fastPtr != null) {
			slowPtr = slowPtr.next;
			fastPtr = fastPtr.next;
		}

		// The slow pointer is now at the Nth node from the end
		return slowPtr.data;
	}
}

public class FindNthFromEnd_3 {
	public static void main(String[] args) {
		LinkedList03 list = new LinkedList03();
		list.addNode(1);
		list.addNode(2);
		list.addNode(3);
		list.addNode(4);
		list.addNode(5);
		list.addNode(6);
		list.addNode(7);
		list.addNode(8);
		list.addNode(9);

		int n = 2;
		int result = list.findNthFromEnd(n);
		System.out.println("Nth node from the end: " + result);
	}
}
```

Question 4

Given a singly linked list of characters, write a function that returns true if the given list is a palindrome, else false.



Examples:

Input: R-A-D-A-R-NULL
Output: Yes
Input: C-O-D-E-NULL
Output: No

Solution Code:

```
package in.ineuron.pptAssignment112;

class Node04 {
    char data;
    Node04 next;

    public Node04(char data) {
        this.data = data;
        next = null;
    }
}

class LinkedList04 {
    Node04 head;

    public void addNode(char data) {
        Node04 newNode = new Node04(data);
        if (head == null) {
            head = newNode;
        } else {
            Node04 current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    public boolean isPalindrome() {
        if (head == null) {
            // Empty list
```

```
                        return true;
                }

                Node04 slowPtr = head;
                Node04 fastPtr = head;

                // Find the middle node of the linked list
                while (fastPtr != null && fastPtr.next != null) {
                        slowPtr = slowPtr.next;
                        fastPtr = fastPtr.next.next;
                }

                // Reverse the second half of the linked list
                Node04 reversedHead = reverse(slowPtr);

                // Compare the reversed second half with the first half
                Node04 firstHalf = head;
                Node04 secondHalf = reversedHead;

                while (secondHalf != null) {
                        if (firstHalf.data != secondHalf.data) {
                                // Not a PALINDROME
                                return false;
                        }
                        firstHalf = firstHalf.next;
                        secondHalf = secondHalf.next;
                }

                // PALINDROME
                return true;
        }

        private Node04 reverse(Node04 node) {
                Node04 prev = null;
                Node04 current = node;
                Node04 next = null;

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

                return prev;
        }
}
```

```
public class Palindrome_4 {
        public static void main(String[] args) {
                LinkedList04 list = new LinkedList04();
                list.addNode('R');
                list.addNode('A');
                list.addNode('D');
                list.addNode('A');
                list.addNode('R');

                boolean isPalindrome = list.isPalindrome();
                System.out.println("Is the linked list a palindrome? :: " + isPalindrome);
        }
}
```

Question 5

Given a linked list of N nodes such that it may contain a loop.
A loop here means that the last node of the link list is connected to the node at position X(1-based index). If the link list does not have any loop, X=0.
Remove the loop from the linked list, if it is present, i.e. unlink the last node which is forming the loop.

Example 1:
Input: N = 3
value[] = {1,3,4}
X = 2
Output:1
Explanation: The link list looks like

```
1 - 3 - 4
    ^   |
    |___|
```

A loop is present. If you remove it successfully, the answer will be 1.

Example 2:
Input :N = 4
value[] = {1,8,3,4}
X = 0
Output:1
Explanation: The Linked list does not contains any loop.

Example 3:
Input: N = 4
value[] = {1,2,3,4}

```
        X = 1
    Output:1
    Explanation: The link list looks like
        1 - 2 - 3 - 4
        ^          |
        |_______|
    A loop is present. If you remove it successfully, the answer will be 1.
```

Solution Code:

```
package in.ineuron.pptAssignment112;

public class RemoveLoop_5 {

    public static void main(String[] args) {
        // Create a linked list with a loop
        int[] value = { 1, 2, 3, 4 };
        Node05 head = new Node05(value[0]);
        Node05 current = head;
        for (int i = 1; i < value.length; i++) {
            current.next = new Node05(value[i]);
            current = current.next;
        }
        current.next = head;

        // Remove the loop
        removeLoop(head);

        // Print the linked list
        printList(head);
    }

    private static void removeLoop(Node05 head) {
        // Find the first node in the loop
        Node05 slow = head;
        Node05 fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
            if (slow == fast) {
                break;
            }
        }

        // If there is no loop, return
        if (fast == null) {
            return;
        }
```

```
			// Find the node that is connected to the first node in the loop
			Node05 runner = slow;
			while (runner != fast) {
				runner = runner.next;
				fast = fast.next;
			}

			// Break the loop by unlinking the node that is connected to the first node in
			// the loop
			runner.next = null;
		}

		private static void printList(Node05 head) {
			Node05 current = head;
			while (current != null) {
				System.out.print(current.data + " ");
				current = current.next;
			}
			System.out.println();
		}

	}

	class Node05 {

		int data;
		Node05 next;

		public Node05(int data) {
			this.data = data;
		}

	}
```

Question 6

Given a linked list and two integers M and N. Traverse the linked list such that you retain M nodes then delete next N nodes, continue the same till end of the linked list.
Difficulty Level: Rookie

Examples:
Input: M = 2, N = 2
Linked List: 1-2-3-4-5-6-7-8
Output:
Linked List: 1-2-5-6

Input: M = 3, N = 2
Linked List: 1-2-3-4-5-6-7-8-9-10
Output:
Linked List: 1-2-3-6-7-8

Input: M = 1, N = 1
Linked List: 1-2-3-4-5-6-7-8-9-10
Output:
Linked List: 1-3-5-7-9

Solution Code:

```
package in.ineuron.pptAssignment112;
public class TraverseLinkedList_6 {

    public static void main(String[] args) {
        // Create a linked list
        Node06 head = new Node06(1);
        head.next = new Node06(2);
        head.next.next = new Node06(3);
        head.next.next.next = new Node06(4);
        head.next.next.next.next = new Node06(5);
        head.next.next.next.next.next = new Node06(6);
        head.next.next.next.next.next.next = new Node06(7);
        head.next.next.next.next.next.next.next = new Node06(8);

        // Traverse the linked list with M = 2 and N = 2
        traverseLinkedList(head, 2, 2);

        // Print the linked list
        printLinkedList(head);
    }

    private static void traverseLinkedList(Node06 head, int m, int n) {
        Node06 current = head;
        Node06 prev = null;
```

```
                while (current != null) {
                        // Retain M nodes
                        for (int i = 0; i < m; i++) {
                                prev = current;
                                current = current.next;
                        }

                        // Delete N nodes
                        for (int i = 0; i < n; i++) {
                                prev.next = current.next;
                                current = current.next;
                        }
                }
        }

        private static void printLinkedList(Node06 head) {
                Node06 current = head;
                while (current != null) {
                        System.out.print(current.data + " ");
                        current = current.next;
                }
                System.out.println();
        }
}

class Node06 {
        int data;
        Node06 next;

        public Node06(int data) {
                this.data = data;
                this.next = null;
        }
}
```

Question 7

Given two linked lists, insert nodes of second list into first list at alternate positions of first list. For example, if first list is 5-7-17-13-11 and second is 12-10-2-4-6, the first list should become 5-12-7-10-17-2-13-4-11-6 and second list should become empty. The nodes of second list should only be inserted when there are positions available. For example, if the first list is 1-2-3 and second list is 4-5-6-7-8, then first list should become 1-4-2-5-3-6 and second list to 7-8.

Use of extra space is not allowed (Not allowed to create additional nodes), i.e., insertion must be done in-place. Expected time complexity is O(n) where n is number of nodes in first list.

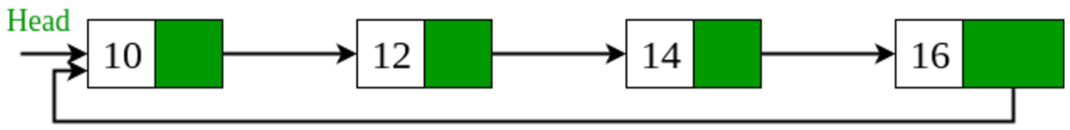
Solution Code:

```
package in.ineuron.pptAssignment112;

public class InsertNodesInPlace_7 {

    public static void main(String[] args) {
        // Create two linked lists
        Node head1 = new Node(5);
        head1.next = new Node(7);
        head1.next.next = new Node(17);
        head1.next.next.next = new Node(13);
        head1.next.next.next.next = new Node(11);

        Node head2 = new Node(12);
        head2.next = new Node(10);
        head2.next.next = new Node(2);
        head2.next.next.next = new Node(4);
        head2.next.next.next.next = new Node(6);

        // Insert nodes of second list into first list at alternate positions
        insertNodesInPlace(head1, head2);

        // Print the first linked list
        printLinkedList(head1);
    }

    private static void insertNodesInPlace(Node head1, Node head2) {
        Node current1 = head1;
        Node current2 = head2;

        while (current1 != null && current2 != null) {
            // Insert the node from the second list after the current node in the first
list
            current1.next = current2;
            current1 = current1.next.next;
            current2 = current2.next;
```

```
                }

                // Set the next node of the last node in the first list to the last node in the
                // second list
                current1.next = current2;
        }

        private static void printLinkedList(Node head) {
                Node current = head;
                while (current != null) {
                        System.out.print(current.data + " ");
                        current = current.next;
                }
                System.out.println();
        }
}

class Node {
        int data;
        Node next;

        public Node(int data) {
                this.data = data;
                this.next = null;
        }
}
```

Question 8

Given a singly linked list, find if the linked list is [circular](https://www.geeksforgeeks.org/circular-linked-list/amp/) or not.
A linked list is called circular if it is not NULL-terminated and all nodes are connected in the form of a cycle. Below is an example of a circular linked list.



Solution Code:

```
package in.ineuron.pptAssignment112;

class Node08 {
	int data;
	Node08 next;

	public Node08(int data) {
		this.data = data;
		next = null;
	}
}

class LinkedList {
	Node08 head;

	public void addNode(int data) {
		Node08 newNode = new Node08(data);
		if (head == null) {
			head = newNode;
		} else {
			Node08 current = head;
			while (current.next != null) {
				current = current.next;
			}
			current.next = newNode;
		}
	}

	public boolean isCircular() {
		if (head == null) {
			// Empty list
			return false;
		}

		Node08 slowPtr = head;
		Node08 fastPtr = head;
```

```
                while (fastPtr != null && fastPtr.next != null) {
                        slowPtr = slowPtr.next;
                        fastPtr = fastPtr.next.next;

                        if (slowPtr == fastPtr) {
                                // Cycle detected
                                return true;
                        }
                }

                // No cycle found
                return false;
        }
}

public class IsCircular_8 {
        public static void main(String[] args) {
                LinkedList list = new LinkedList();
                list.addNode(1);
                list.addNode(2);
                list.addNode(3);
                list.addNode(4);
                list.addNode(5);

                // Create a cycle by connecting the tail to the second node
                list.head.next.next.next.next.next = list.head.next;

                if (list.isCircular()) {
                        System.out.println("The linked list is circular.");
                } else {
                        System.out.println("The linked list is not circular.");
                }
        }
}
```

GITHUB: https://github.com/devavratwadekar/ineuron_ppt_ProgramAssignmentCode