

```
class Node {
    int data;
    Node next;

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

class SinglyLinkedList {
    Node head;

    public void insertAtPosition(int data, int position) {
        Node newNode = new Node(data);

        if (position == 1) {
            newNode.next = head;
            head = newNode;
            return;
        }

        Node temp = head;
        int currentPos = 1;

        while (temp != null && currentPos < position - 1) {
            temp = temp.next;
            currentPos++;
        }

        if (temp == null)
            return;

        newNode.next = temp.next;
        temp.next = newNode;
    }

    public void display() {
        Node temp = head;
        while (temp != null) {
```

```
        System.out.print(temp.data);
        if (temp.next != null) System.out.print(" -> ");
        temp = temp.next;
    }
    System.out.println();
}

public class SinglyLinkedListInsertion {
    public static void main(String[] args) {
        SinglyLinkedList list = new SinglyLinkedList();

        list.insertAtPosition(10, 1);
        list.insertAtPosition(20, 2);
        list.insertAtPosition(30, 3);
        list.insertAtPosition(40, 4);

        System.out.println("Initial List:");
        list.display();

        java.util.Scanner sc = new java.util.Scanner(System.in);

        System.out.print("Enter data to insert: ");
        int data = sc.nextInt();

        System.out.print("Enter position: ");
        int position = sc.nextInt();

        list.insertAtPosition(data, position);

        System.out.println("Updated List:");
        list.display();

        sc.close();
    }
}
```

```
PS E:\JAVA PROGRAMS\steparyansingh\year2\dsa\week13\assignment> javac SinglyLinkedListInsertion.java
>> java SinglyLinkedListInsertion
Initial List:
10 -> 20 -> 30 -> 40
Enter data to insert: 45
Enter position: 5
Updated List:
10 -> 20 -> 30 -> 40 -> 45
```

```
class Node {
    int data;
    Node next;

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

class SinglyLinkedList {
    Node head;

    public boolean detectLoop() {
        Node slow = head;
        Node fast = head;

        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;

            if (slow == fast) {
                removeLoop(slow);
                return true;
            }
        }
        return false;
    }

    void removeLoop(Node node) {
        Node temp = node;
        while (temp.next != node) {
            temp = temp.next;
        }
        temp.next = null;
    }
}
```

```
private void removeLoop(Node loopNode) {
    Node ptr1 = head;
    Node ptr2 = loopNode;

    while (ptr1 != ptr2) {
        ptr1 = ptr1.next;
        ptr2 = ptr2.next;
    }

    Node temp = ptr2;
    while (temp.next != ptr1) {
        temp = temp.next;
    }

    temp.next = null;
}

public void display() {
    Node temp = head;
    while (temp != null) {
        System.out.print(temp.data);
        if (temp.next != null) System.out.print(" -> ");
        temp = temp.next;
    }
    System.out.println();
}

public class LoopRemovalInSinglyLinkedList {
    public static void main(String[] args) {

        SinglyLinkedList list = new SinglyLinkedList();

        list.head = new Node(10);
        list.head.next = new Node(20);
        list.head.next.next = new Node(30);
        list.head.next.next.next = new Node(40);
        list.head.next.next.next.next = new Node(50);
    }
}
```

```

        list.head.next.next.next.next = list.head.next.next;

        System.out.println("Loop detection in progress...");
        boolean found = list.detectLoop();

        if (found)
            System.out.println("Loop detected and removed");
        else
            System.out.println("No loop found");

        System.out.println("List after loop removal:");
        list.display();
    }
}

```

```

PS E:\JAVA PROGRAMS\steparyansingh\year2\dsa\week13\assignment> javac LoopRemovalInSinglyLinkedList.java
>> java LoopRemovalInSinglyLinkedList
>>
Loop detection in progress...
Loop detected and removed
List after loop removal:
10 -> 20 -> 30 -> 40 -> 50

```

```

class Node {
    int data;
    Node prev, next;

    Node(int data) {

```

```
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}

class DoublyLinkedList {
    Node head;

    public void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.next != null)
            temp = temp.next;
        temp.next = newNode;
        newNode.prev = temp;
    }

    public void deleteAll(int value) {
        Node temp = head;
        while (temp != null) {
            if (temp.data == value) {
                if (temp.prev != null)
                    temp.prev.next = temp.next;
                else
                    head = temp.next;

                if (temp.next != null)
                    temp.next.prev = temp.prev;
            }
            temp = temp.next;
        }
    }

    public void display() {
        Node temp = head;
```

```

        while (temp != null) {
            System.out.print(temp.data);
            if (temp.next != null) System.out.print(" <-> ");
            temp = temp.next;
        }
        System.out.println();
    }

}

public class DeleteAllOccurrencesDLL {
    public static void main(String[] args) {

        DoublyLinkedList list = new DoublyLinkedList();

        list.insert(10);
        list.insert(20);
        list.insert(30);
        list.insert(20);
        list.insert(40);

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

        list.deleteAll(20);

        System.out.println("List after deleting all occurrences of 20:");
        list.display();
    }
}

```

```

PS E:\JAVA PROGRAMS\steparyansingh\year2\dsa\week13\assignment> javac DeleteAllOccurrencesDLL.java
>> java DeleteAllOccurrencesDLL
>>
Original List:
10 <-> 20 <-> 30 <-> 20 <-> 40
List after deleting all occurrences of 20:
10 <-> 30 <-> 40

```

```
class Node {
```

```
int data;
Node prev, next;

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

class DoublyLinkedList {
    Node head;

    public void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            return;
        }
        Node temp = head;
        while (temp.next != null)
            temp = temp.next;
        temp.next = newNode;
        newNode.prev = temp;
    }

    public void bubbleSort() {
        if (head == null)
            return;
        boolean swapped;
        Node temp;
        do {
            swapped = false;
            temp = head;
            while (temp.next != null) {
                if (temp.data > temp.next.data) {
                    int t = temp.data;
                    temp.data = temp.next.data;
                    temp.next.data = t;
                    swapped = true;
                }
            }
        } while (swapped);
    }
}
```

```
        }
        temp = temp.next;
    }
} while (swapped);
}

public void display() {
    Node temp = head;
    while (temp != null) {
        System.out.print(temp.data);
        if (temp.next != null) System.out.print(" <-> ");
        temp = temp.next;
    }
    System.out.println();
}

public class SortDLLBubble {
    public static void main(String[] args) {

        DoublyLinkedList list = new DoublyLinkedList();

        list.insert(40);
        list.insert(10);
        list.insert(30);
        list.insert(20);

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

        list.bubbleSort();

        System.out.println("Sorted List:");
        list.display();
    }
}
```

```
PS E:\JAVA PROGRAMS\steparyansingh\year2\dsa\week13\assignment> javac SortDLLBubble.java
>> java SortDLLBubble
● >>
Original List:
40 <-> 10 <-> 30 <-> 20
Sorted List:
10 <-> 20 <-> 30 <-> 40
```