Linked List

Assignment 1

```
class Node {

    private String data;
    private Node next;

    public Node(String data) {
        this.data = data;
    }

    public void setData(String data) {
        this.data = data;
    }

    public void setNext(Node node) {
        this.next = node;
    }

    public String getData() {
        return this.data;
    }

    public Node getNext() {
        return this.next;
    }
```

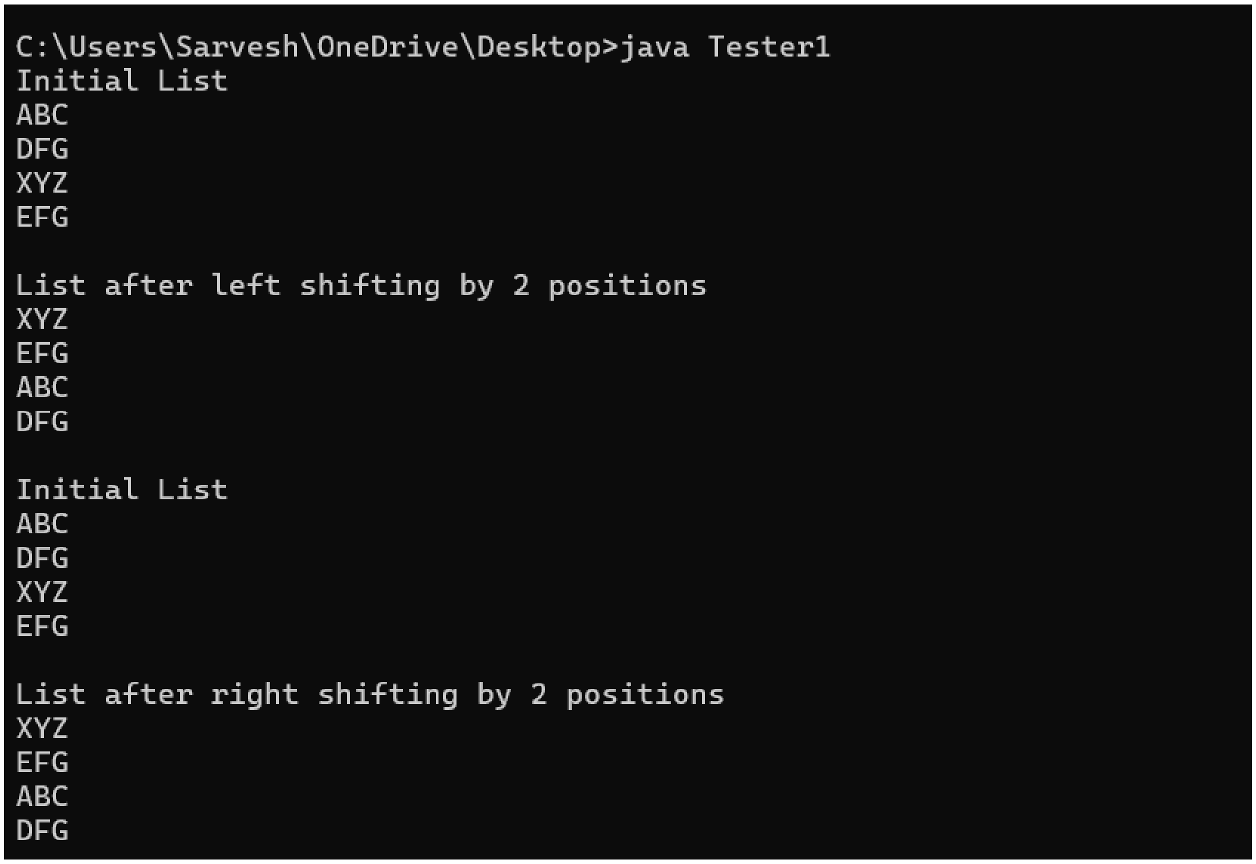
```
}


class LinkedList {

    private Node head;
    private Node tail;

    public Node getHead() {
        return this.head;
    }

    public Node getTail() {
        return this.tail;
    }

    public void setHead(Node head) {
        this.head = head;
    }

    public void setTail(Node tail) {
        this.tail = tail;
    }

    public void addAtEnd(String data) {
        Node node = new Node(data);
        if (this.head == null) {
            this.head = this.tail = node;
        } else {
            this.tail.setNext(node);
            this.tail = node;
        }
    }

    public void addAtBeginning(String data) {
        Node node = new Node(data);

        if (this.head == null) {
            this.head = this.tail = node;
        } else {
            node.setNext(this.head);
            this.head = node;
        }
    }

    public void display() {
        Node temp = this.head;

        while (temp != null) {
            System.out.println(temp.getData());
            temp = temp.getNext();
        }
    }
```

```
}

public Node find(String data) {
    Node temp = this.head;

    while (temp != null) {
        if (temp.getData().equals(data))
            return temp;
        temp = temp.getNext();
    }
    return null;
}

public void insert(String data, String dataBefore) {
    Node node = new Node(data);

    if (this.head == null)
        this.head = this.tail = node;
    else {
        Node nodeBefore = this.find(dataBefore);
        if (nodeBefore != null) {
            node.setNext(nodeBefore.getNext());
            nodeBefore.setNext(node);
            if (nodeBefore == this.tail)
                this.tail = node;
        } else
            System.out.println("Node not found");
    }
```

```
}

public void delete(String data) {

    if (this.head == null)
        System.out.println("List is empty");
    else {
        Node node = this.find(data);

        if (node == null)
            System.out.println("Node not found");

        if (node == this.head) {
            this.head = this.head.getNext();
            node.setNext(null);

            if (node == this.tail)
                tail = null;
        } else {
            Node nodeBefore = null;
            Node temp = this.head;
            while (temp != null) {
                if (temp.getNext() == node) {
                    nodeBefore = temp;
                    break;
                }
                temp = temp.getNext();
                }

                nodeBefore.setNext(node.getNext());

                if (node == this.tail)
                    this.tail = nodeBefore;
                node.setNext(null);
            }
        }
    }
}


class Tester1 {

    public static void main(String args[]) {
        LinkedList linkedList1 = new LinkedList();
        linkedList1.addAtEnd("ABC");
        linkedList1.addAtEnd("DFG");
        linkedList1.addAtEnd("XYZ");
        linkedList1.addAtEnd("EFG");

        LinkedList linkedList2 = new LinkedList();
        linkedList2.addAtEnd("ABC");
        linkedList2.addAtEnd("DFG");
        linkedList2.addAtEnd("XYZ");
        linkedList2.addAtEnd("EFG");
        System.out.println("Initial List");
        linkedList1.display();

        System.out.println("\nList after left shifting by 2 positions");
        shiftListLeft(linkedList1, 2);
        linkedList1.display();

        System.out.println("\nInitial List");
        linkedList2.display();

        System.out.println("\nList after right shifting by 2 positions");
        shiftListRight(linkedList2, 2);
        linkedList2.display();
    }

    public static void shiftListLeft(LinkedList linkedList, int n) {
        if (linkedList.getHead() == null || n <= 0) return;

        int length = getLength(linkedList);
        n = n % length;
        if (n == 0) return;

        Node current = linkedList.getHead();
        Node prevTail = linkedList.getTail();

        for (int i = 1; i < n; i++) {
```

```
        current = current.getNext();
    }

    Node newHead = current.getNext();
    current.setNext(null);
    prevTail.setNext(linkedList.getHead());
    linkedList.setHead(newHead);
}

public static void shiftListRight(LinkedList linkedList, int n) {
    if (linkedList.getHead() == null || n <= 0) return;

    int length = getLength(linkedList);
    n = n % length;
    if (n == 0) return;

    shiftListLeft(linkedList, length - n);
}

private static int getLength(LinkedList linkedList) {
    int length = 0;
    Node current = linkedList.getHead();
    while (current != null) {
        length++;
        current = current.getNext();
    }
    return length;
    }
}
```

Output-



```
C:\Users\Sarvesh\OneDrive\Desktop>java Tester1
Initial List
ABC
DFG
XYZ
EFG

List after left shifting by 2 positions
XYZ
EFG
ABC
DFG

Initial List
ABC
DFG
XYZ
EFG

List after right shifting by 2 positions
XYZ
EFG
ABC
DFG
```