Linked List

Assignment 2

```
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 Tester2 {

    public static void main(String args[]) {

        LinkedList linkedList = new LinkedList();
        LinkedList reversedLinkedList = new LinkedList();

        linkedList.addAtEnd("Data");
        linkedList.addAtEnd("Structures");
        linkedList.addAtEnd("and");
        linkedList.addAtEnd("Algorithms");

        System.out.println("Initial List");
        linkedList.display();
```

```
        System.out.println();

        reverseList(linkedList.getHead(), reversedLinkedList);
        System.out.println("Reversed List");
        reversedLinkedList.display();
    }

    public static void reverseList(Node head, LinkedList reversedLinkedList) {
        if (head == null) {
            return;
        }
        reverseList(head.getNext(), reversedLinkedList);
        reversedLinkedList.addAtEnd(head.getData());
    }
}
```

Output-

```
C:\Users\Sarvesh\OneDrive\Desktop>java Tester2
Initial List
Data
Structures
and
Algorithms

Reversed List
Algorithms
and
Structures
Data
```