Linked List Unit II
Q1 Delete node
You have been given a linked list of integers. Your task is to write a function that deletes a node from a given position, 'POS'.
Note :
Assume that the Indexing for the linked list always starts from 0.

Sample Input 1 :
```
3 4 5 2 6 1 9 -1
3
```
Sample Output 1 :
```
3 4 5 6 1 9
```

Input
```
10 20 30 40 50 60 70 -1
4
```
Output
```
10 20 30 40 60 70
```

```java
import java.util.Scanner;

class Node {
    int data;
    Node next;

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

class LinkedList {
    private Node head;

    // Method to insert data into the linked list
    void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }
    // Method to delete a node at a given position
    void deleteNode(int pos) {
        if (head == null || pos < 0) {
            return; // If list is empty or position is invalid, do nothing
        }

        if (pos == 0) { // Delete the head node
            head = head.next;
            return;
        }

        Node temp = head;
        for (int i = 0; temp != null && i < pos - 1; i++) {
            temp = temp.next;
        }

        if (temp == null || temp.next == null) {
            return; // If position is greater than the number of nodes, do nothing
        }

        temp.next = temp.next.next; // Delete the node
    }

    // Method to display the linked list
    void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList list = new LinkedList();

        // Read linked list input until -1 is encountered
        while (true) {
            int value = scanner.nextInt();
            if (value == -1) {
                break;
            }
            list.insert(value);
        }
        // Read the position to delete
        int pos = scanner.nextInt();
        scanner.close();

        // Delete the node at position pos
        list.deleteNode(pos);

        // Display the modified linked list
        list.display();
    }
}
```

Q2 Insert Node

You have been given a linked list of integers. Your task is to write a function that inserts a node at a given position, 'pos'.

Note:

Assume that the Indexing for the linked list always starts from 0.

Input

```
3 4 5 2 6 1 9 -1
3
100
```

Sample Output 1:

```
3 4 5 100 2 6 1 9
```

Input

```
10 20 30 40 50 -1
2
25
```

Output

```
10 20 25 30 40 50
```

Input

```
10 20 30 40 50 60 -1
0
9
```

Output

```
9 10 20 30 40 50 60
```

```
import java.util.Scanner;

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

class LinkedList {
    private Node head;

    // Method to insert data into the linked list
    void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }

    // Method to insert a node at a given position
    void insertAtPosition(int pos, int data) {
        Node newNode = new Node(data);

        if (pos == 0) { // Insert at head
            newNode.next = head;
            head = newNode;
            return;
        }

        Node temp = head;
        for (int i = 0; temp != null && i < pos - 1; i++) {
            temp = temp.next;
        }

        if (temp == null) {
            return; // Position is greater than the list length, do nothing
        }

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

    // Method to display the linked list
    void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}

class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList list = new LinkedList();

        // Read linked list input until -1 is encountered
        while (true) {
            int value = scanner.nextInt();
            if (value == -1) {
                break;
            }
            list.insert(value);
        }

        // Read position and value to be inserted
        int pos = scanner.nextInt();
        int data = scanner.nextInt();
        scanner.close();

        // Insert the node at the given position
        list.insertAtPosition(pos, data);

        // Display the modified linked list
        list.display();
    }
}
```

Q3 AppendLastNToFirst You have been given a singly linked list of integers along with an integer 'N'. Write a function to append the last 'N' nodes towards the front of the singly linked list and returns the new head to the list.

Input

```
1 2 3 4 5 -1
3
```

OUTPUT

```
3 4 5 1 2
```

Input

```
10 20 30 40 50 -1
```

2
Output
40 50 10 20 30

Input
1 2 30 40 50 60 7 8 -1
3
60 7 8 1 2 30 40 50

```
import java.util.Scanner;

class Node {
    int data;
    Node next;

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

class LinkedList {
    private Node head;

    // Method to insert data into the linked list
    void insert(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            temp.next = newNode;
        }
    }

    // Method to append the last N nodes to the front
    void appendLastNToFirst(int n) {
        if (head == null || n <= 0) {
            return;
        }

        // Step 1: Determine the length of the list
        Node temp = head;
        int length = 1;
        while (temp.next != null) {
            temp = temp.next;
            length++;
        }

        // If n is greater than or equal to the length of the list, no change is needed
        if (n >= length) {
            return;
        }

        // Step 2: Identify the (length-n)th node
        int splitPoint = length - n;
        Node newTail = head;
        for (int i = 1; i < splitPoint; i++) {
            newTail = newTail.next;
        }

        // Step 3: Rearrange pointers
        Node newHead = newTail.next;
        temp.next = head; // 'temp' is currently the last node
        newTail.next = null;
        head = newHead;
    }

    // Method to display the linked list
    void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}

class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList list = new LinkedList();

        // Read linked list input until -1 is encountered
        while (true) {
            int value = scanner.nextInt();
            if (value == -1) {
                break;
            }
            list.insert(value);
        }
        // Read the value of N
        int n = scanner.nextInt();
        scanner.close();

        // Append the last N nodes to the front
        list.appendLastNToFirst(n);

        // Display the modified linked list
        list.display();
    }
}
```