Linked list

Creating a singly , doubly linked list. Insertion, deletion, deleting from the middle

Code:

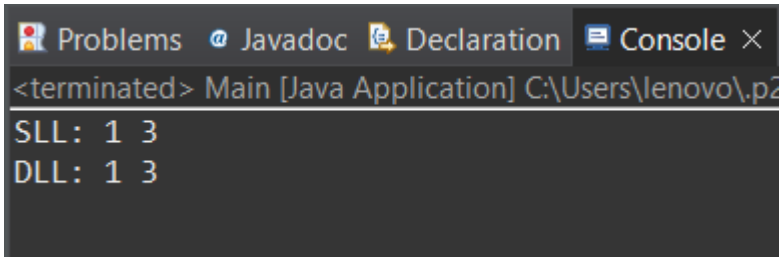
```
class SinglyLinkedList {
    class Node {
        int data;
        Node next;
        Node(int data) {
            this.data = data;
            next = null;
        }
    }

    Node head;

    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;
        }
    }

    void deleteFromMiddle(int pos) {
        if (head == null) return;
        if (pos == 0) {
            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;
        temp.next = temp.next.next;
    }

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

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

    Node head;
```

```
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;
        newNode.prev = temp;
    }
}

void deleteFromMiddle(int pos) {
    if (head == null) return;
    Node temp = head;
    for (int i = 0; temp != null && i < pos; i++) {
        temp = temp.next;
    }
    if (temp == null) return;
    if (temp.next != null) temp.next.prev = temp.prev;
    if (temp.prev != null) temp.prev.next = temp.next;
    else head = temp.next;
}

void printList() {
    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) {
        SinglyLinkedList sll = new SinglyLinkedList();
        sll.insert(1);
        sll.insert(2);
        sll.insert(3);
        sll.deleteFromMiddle(1);
        System.out.print("SLL: ");
        sll.printList();

        DoublyLinkedList dll = new DoublyLinkedList();
        dll.insert(1);
        dll.insert(2);
        dll.insert(3);
        dll.deleteFromMiddle(1);
        System.out.print("DLL: ");
        dll.printList();
    }
}
```

Output:



Time Complexity:

Insertion, Deletion from middle and Print list for both SLL and DLL are O(n)