Algorithm for inserting a node in a singly linked list

**Algorithm** `insert` (newNode, predecessor)

**In:** New node to be inserted after predecessor.

**Out:** {Insert newNode in linked list after predecessor; newNode is inserted at the front of the list if predecessor is null.}

```java
if predecessor = null then {
    newNode.setNext(front)
    front = newNode
}
else {
    newNode.setNext(predecessor.getNext())
    predecessor.setNext(newNode)
}
```
public void insert (LinearNode<T> newNode, LinearNode<T> predecessor) {
    if (predecessor == null) {
        newNode.setNext(front);
        front = newNode;
    } else {
        newNode.setNext(predecessor.getNext());
        predecessor.setNext(newNode);
    }
}
/* Method to add an element to the front of the linked list*/
public void addToFront(T element) {
    LinearNode<T> newNode = new LinearNode<>(element);
    if (isEmpty()) { front = newNode; }
    else {
        newNode.setNext(front);
        front = newNode;
    }
}
/* Method to add an element to the rear of the linked list*/
public void addToRear(T element) {
    LinearNode<T> newNode = new LinearNode<>((element);
    if (isEmpty()) { front = newNode; } 
    else {
        LinearNode<T> current = front;
        while (current.getNext() != null) {
            current = current.getNext();
        }
        current.setNext(newNode);
    }
}
Algorithm for deleting a node from a singly linked list

Algorithm delete (nodeToDelete)
In: node to delete
Out: true if the node was deleted, false otherwise

current = front
predecessor = null
while (current != null) and (current != nodeToDelete) do {
    predecessor = current
    current = current.getNext()
}
if current = null then return false
else {
    if predecessor != null then
        predecessor.setNext(current.getNext())
    else front = front.getNext()
    return true
}
public boolean delete (LinearNode<T> nodeToDelete) {
    LinearNode<T> current, predecessor;
    current = front;
    predecessor = null;
    while ((current != null) && (current != nodeToDelete)) {
        predecessor = current;
        current = current.getNext();
    }
    if (current == null) return false;
    else {
        if (predecessor != null)
            predecessor.setNext(current.getNext());
        else front = front.getNext();
        return true;
    }
}
Doubly Linked List

Node object

prev  element  next

front  tail
Java Class for a Node of a Doubly Linked List

public class LinearNodeDLL<T> {
    private LinearNodeDLL<T> next;
    private LinearNodeDLL<T> prev;
    private T element;

    public LinearNode( ) {
        next = null;
        prev = null;
        element = null;
    }

    public LinearNode (T dataItem) { 
        next = null;
        prev = null;
        element = dataItem;
    }
}