Recall: ListNode class

```java
public class ListNode {
    public String data;
    public ListNode next;

    public ListNode(String d) {
        data = d;
        next = null;
    } //constructor
}
```

Recall: LinkedList class

```java
public class LinkedList {
    private ListNode first = new ListNode(null);
    private ListNode last = first;
    private int length = 0;
    ...
}
```

Detour: Does first.next refer to the `next` reference itself, or to whatever first.next is pointing to? It depends on whether it's on the left or right side of the = operator!

When first.next is on the left side of =

```
first.next = ...; //first.next refers to the first.next reference itself!
```

When first.next is on the right side of =

```
... = first.next; //first.next refers to the node first.next points to!
```
Understanding **append**

```java
public void append(String d) { //place new data at the end of the linked list ...
} //append
```

1) Create a **new node**: `ListNode n = new ListNode(d);`

   ![Image of creating a new node](image)

2) Place the new node at the **end** of the list: `last.next = n;`

   ![Image of placing the new node at the end](image)

3) Update the **last** reference (since the **new node** is now the **last** node): `last = n;`

   ![Image of updating the last reference](image)

4) Update the **length** of the list (since we’ve just **added** a node): `++length;` 

   /*
   * final append code
   */

```java
public void append(String d) { //place new data at the end of the linked list
    ListNode n = new ListNode(d); //Step 1
    last.next = n; //Step 2
    last = n; //Step 3
    ++length; //Step 4
} //append
```
Understanding **prepend**

```java
public void prepend(String d) { //place new data at the **beginning** of the linked list
    //prepend
}
```

1) Create a **new node**: `ListNode n = new ListNode(d);`

2) Place the new node **before** the node after the **first (dummy)** node: `n.next = first.next;`

3) Place the new node **after** the **first (dummy)** node: `first.next = n;`

4) Update the length of the list (since we’ve just added a node): `++length;`

5) As you can see in the case of the **Empty List** in **Step 3**, the **last** reference needs to be updated. If `first` and `last` point to the **same node**, the **last** reference needs to be updated: `if (first == last) last = n;`

    ```java
    public void prepend(String d) { //place new data at the **beginning** of the linked list
        ListNode n = new ListNode(d); //Step 1
        n.next = first.next; //Step 2
        first.next = n; //Step 3
        ++length; //Step 4
        if (first == last) last = n; //Step 5 (can be done any time after Step 1)
    } //prepend
    ```