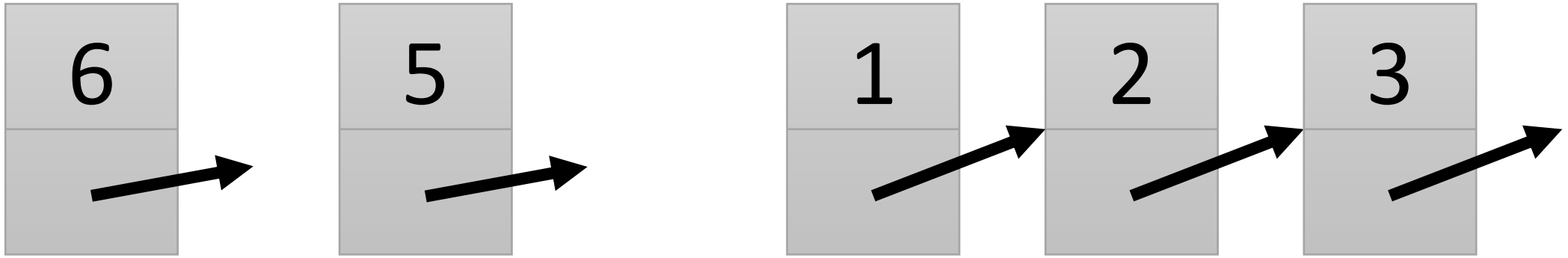
Lab 4: SortedList

CSCI 162 – Introduction to Programming II

Professor William Killian

Primer: Linked Lists

- **Linked Lists** are a collection of **Nodes**
- **Nodes** contain *data* and a *link*
- One *node* is said to *link* to another *node* by setting its *link* field

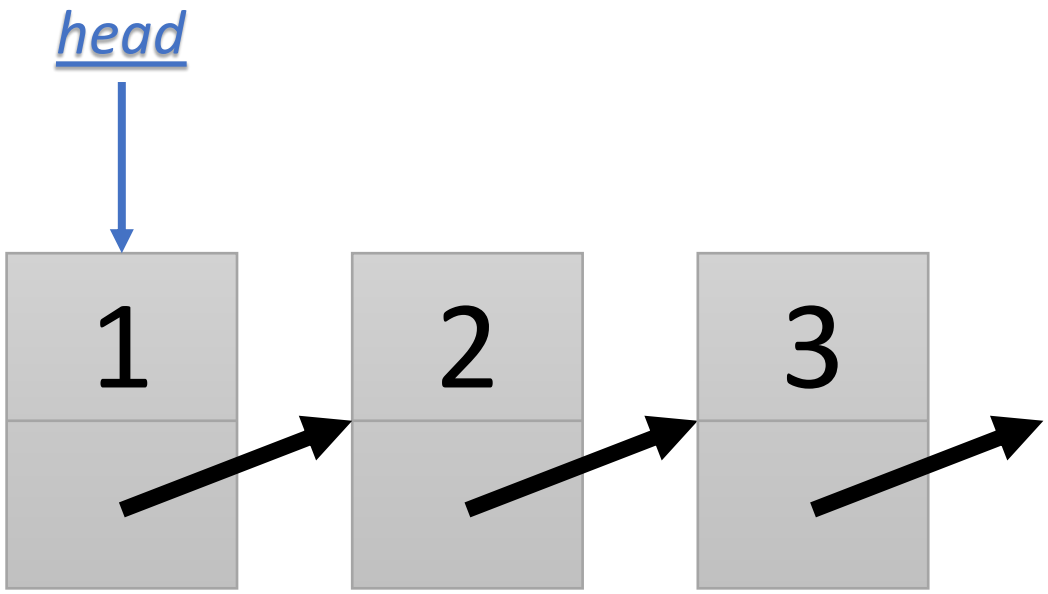


Two separate Nodes

A Linked List comprised of three Nodes

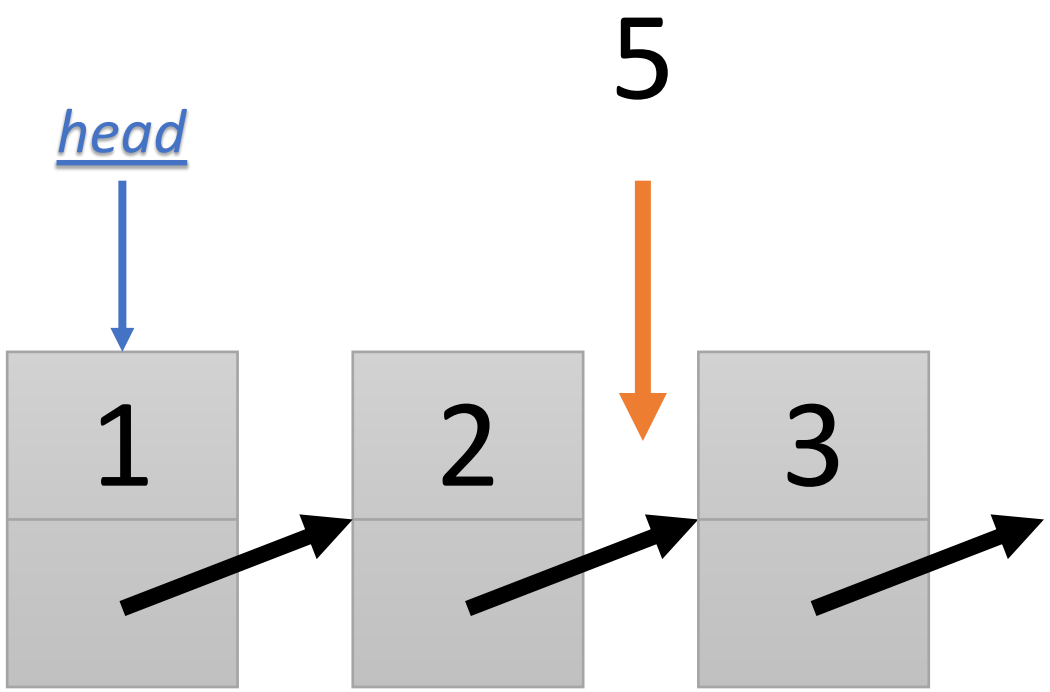
Linked Lists

- Linked Lists always contain a head
- A head is the beginning of the Linked List
- If a head is **null**, then the Linked List is “empty”



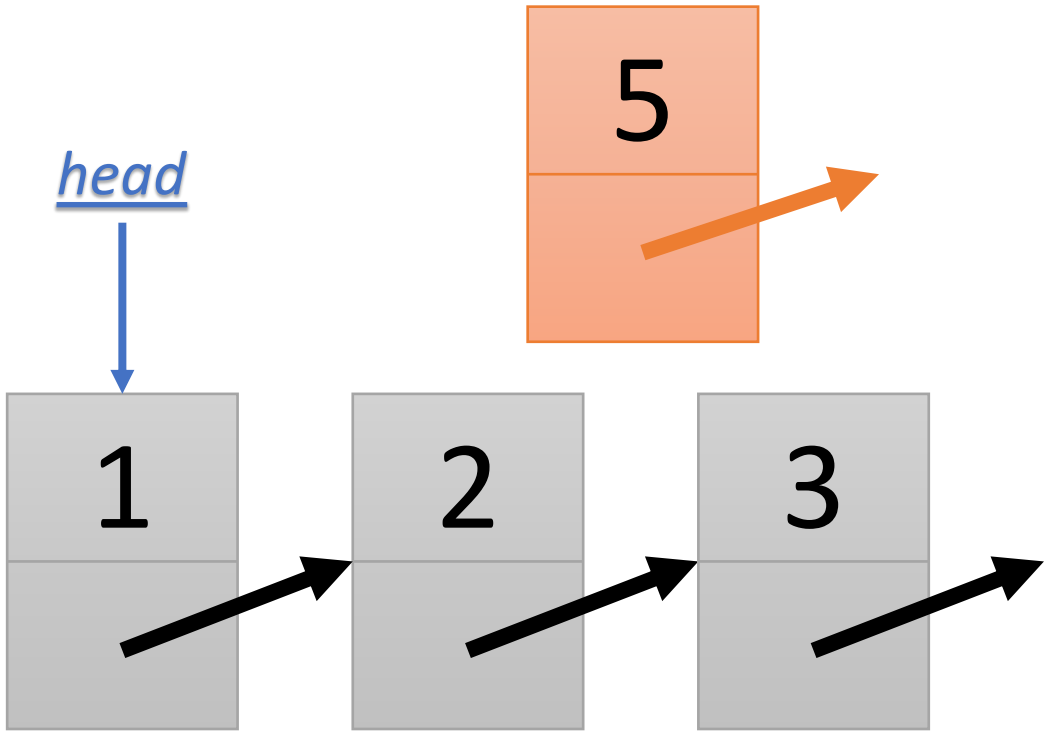
Insertion into Linked Lists

- Suppose we wanted to insert a ”5” between the ”2” and “3” nodes



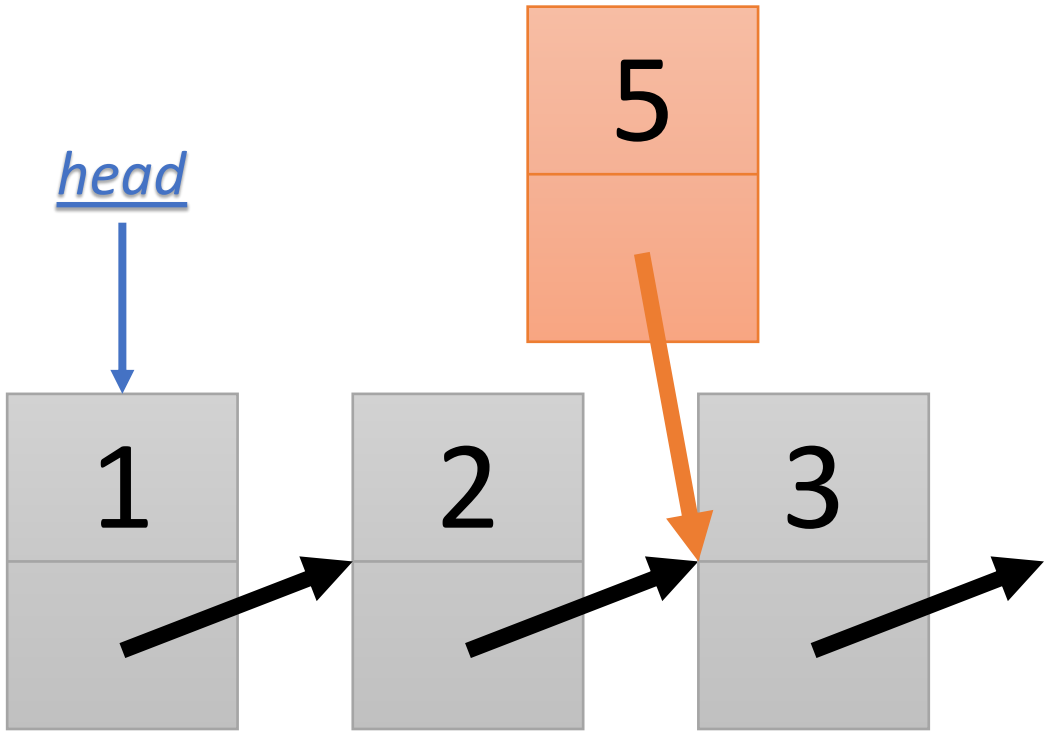
Insertion into Linked Lists

- We first create a new Node for the 5



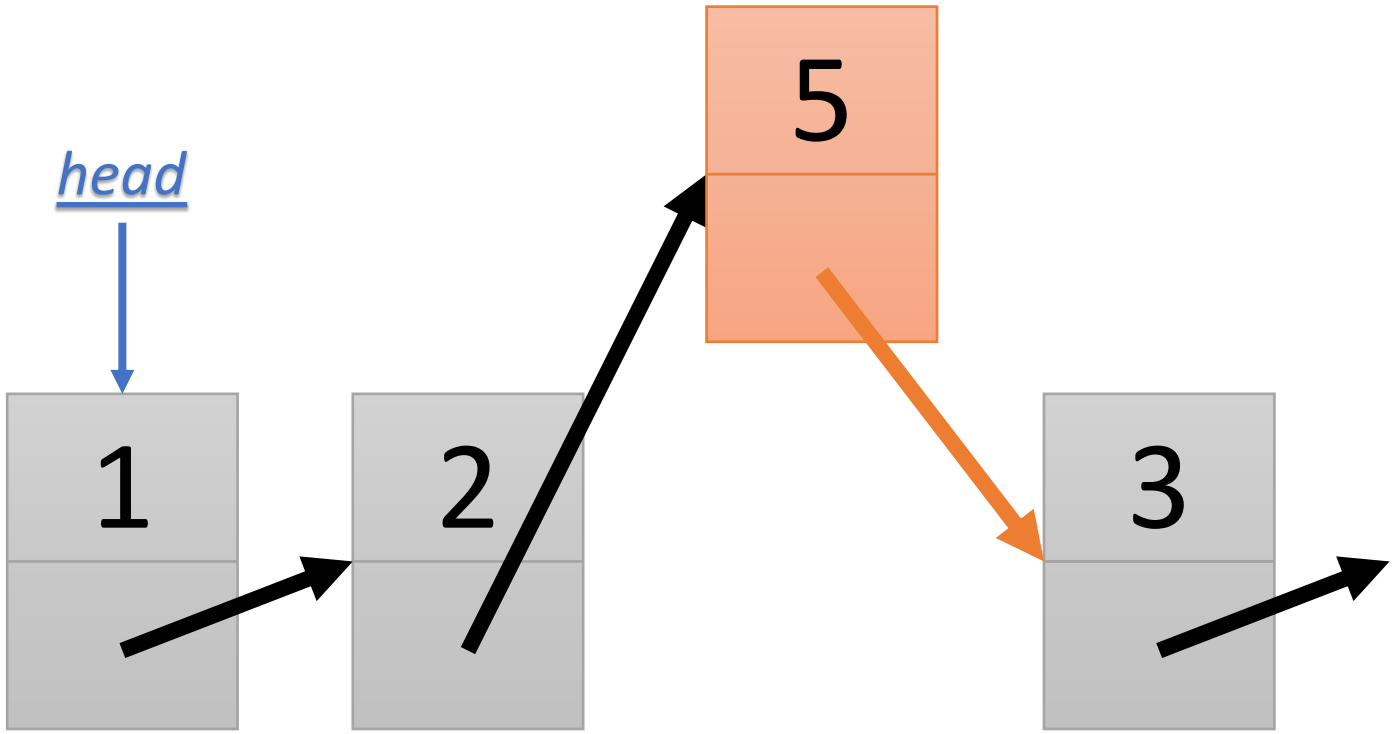
Insertion into Linked Lists

- Then we set its link to be whatever ”2”’s link was



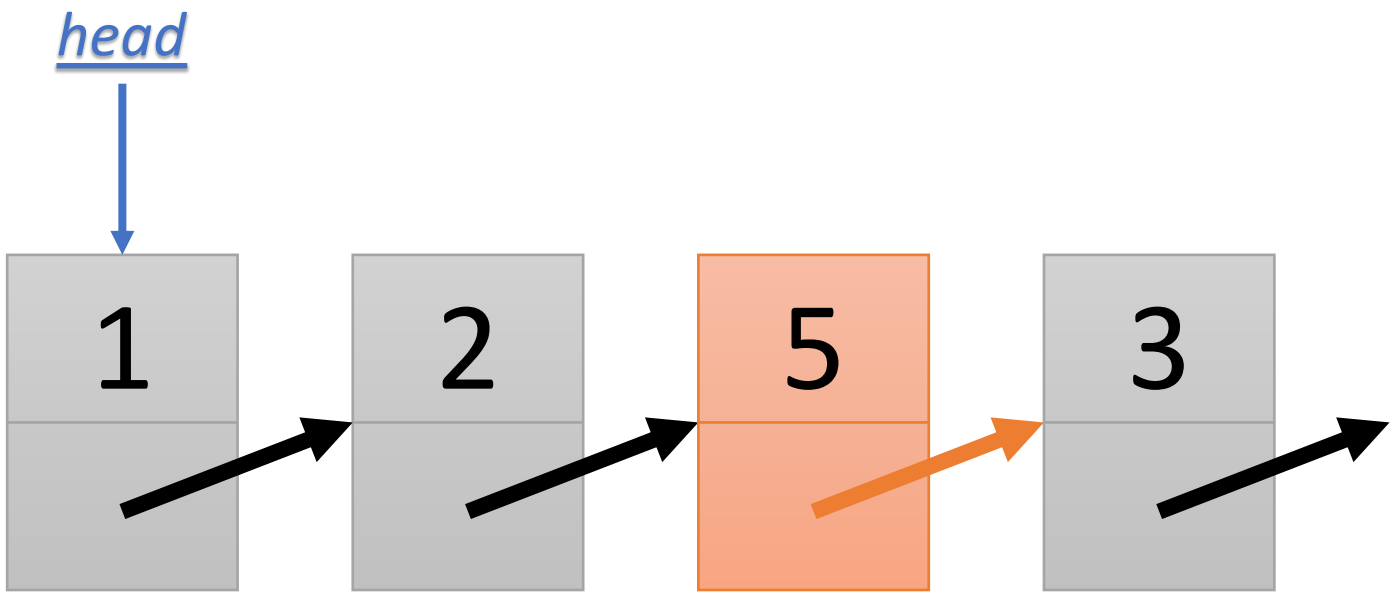
Insertion into Linked Lists

- We then change “2”’s link to our “5” node



Insertion into Linked Lists

- And we have our node inserted



Insertion Notes

- In order for us to insert, we need to have a *Previous Node*
- If we do not have a *Previous Node*, then we insert at the *beginning*

- Consider the following cases:
  - `head == null`
  - `prev == null`
  - `prev != null`

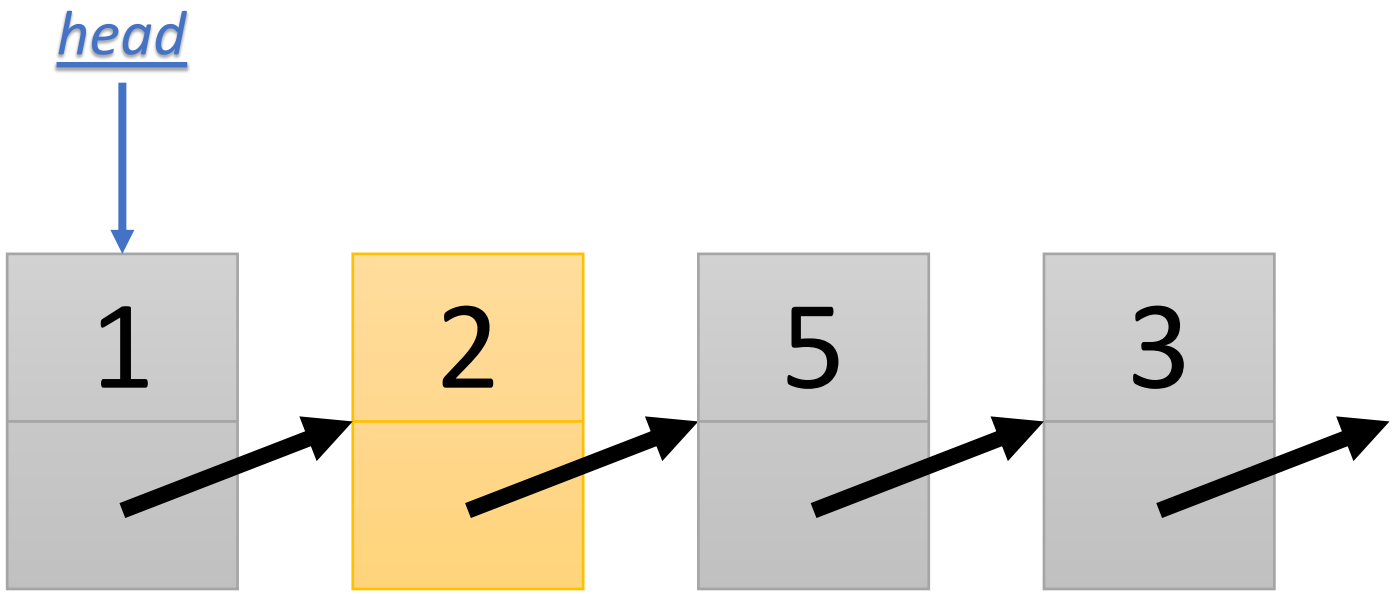
Wait… What's `prev`?

- For this lab, we don't have prev
- Instead, we are implementing a method, `getPredecessor()`

```
/**
 * @return the node which comes before a node with value "v"
 *      In the case where there is no such node, this method
 *      will return null (and should be added to the front)
 */
private DoubleNode getPredecessor (double v) {
  // your code here
}
```

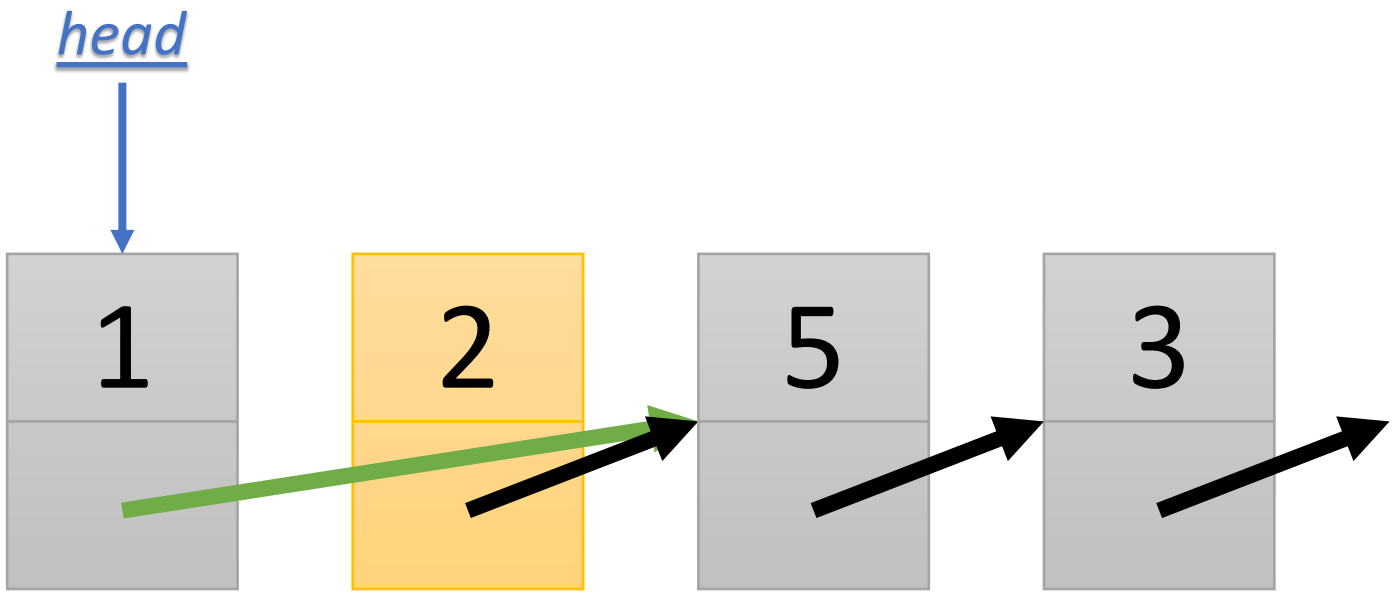
Removal from SortedList

- Removing takes an index
- The Node prior to the index-th node is our prev
- Example: removeAt (1)



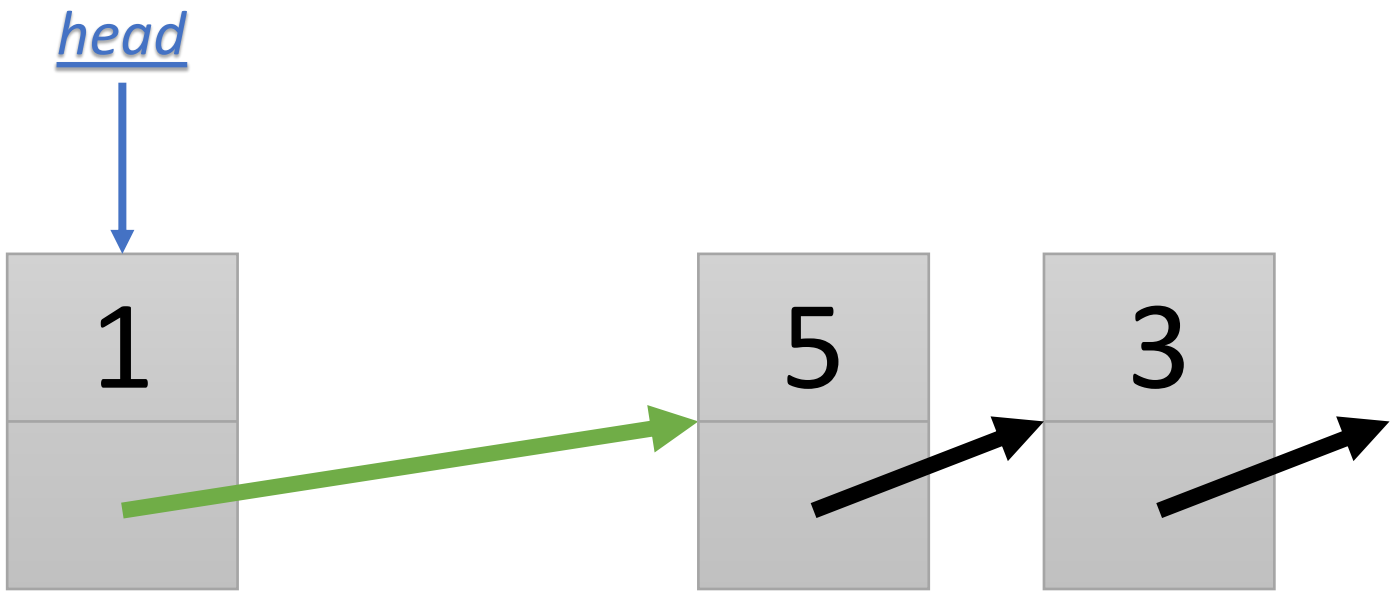
Removal from SortedList

- Removing takes an index
- The Node prior to the index-th node is our prev
- Example: removeAt (1)



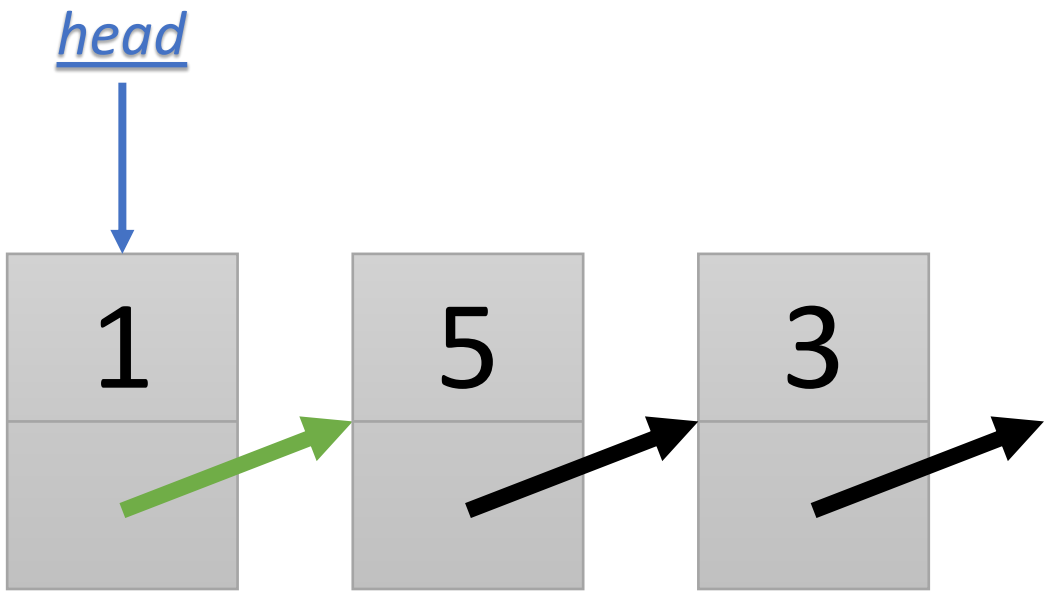
Removal from SortedList

- Removing takes an index
- The Node prior to the index-th node is our prev
- Example: removeAt (1)



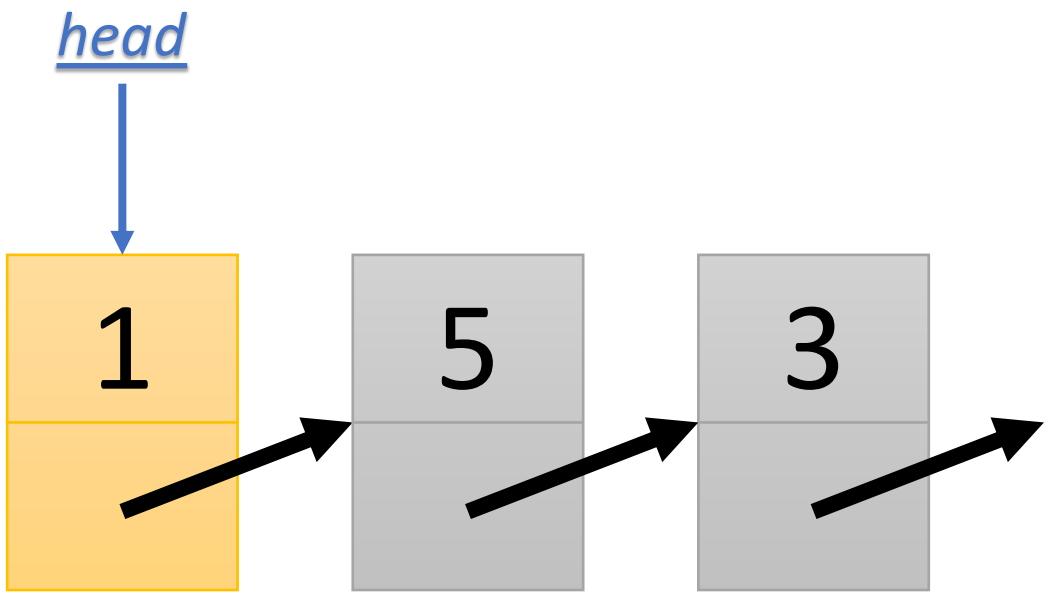
Removal from SortedList

- Removing takes an index
- The Node prior to the index-th node is our prev
- Example: removeAt (1)



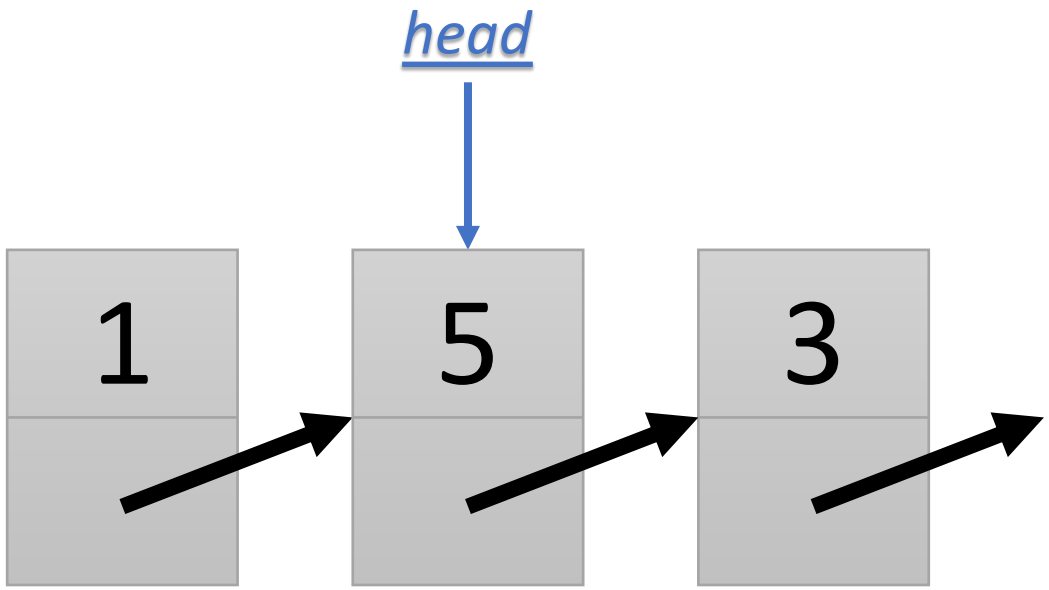
Removal from SortedList

- Example 2: removeAt (0)



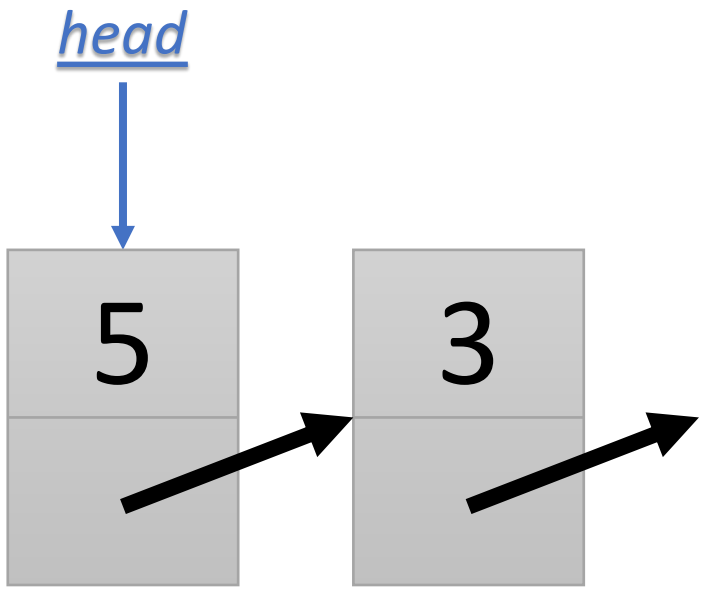
Removal from SortedList

- Example 2: removeAt (0)



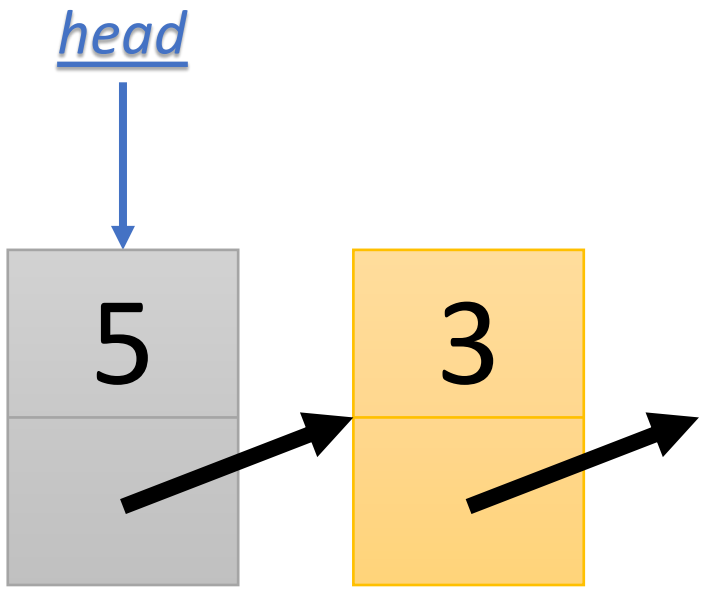
Removal from SortedList

- Example 2: removeAt (0)



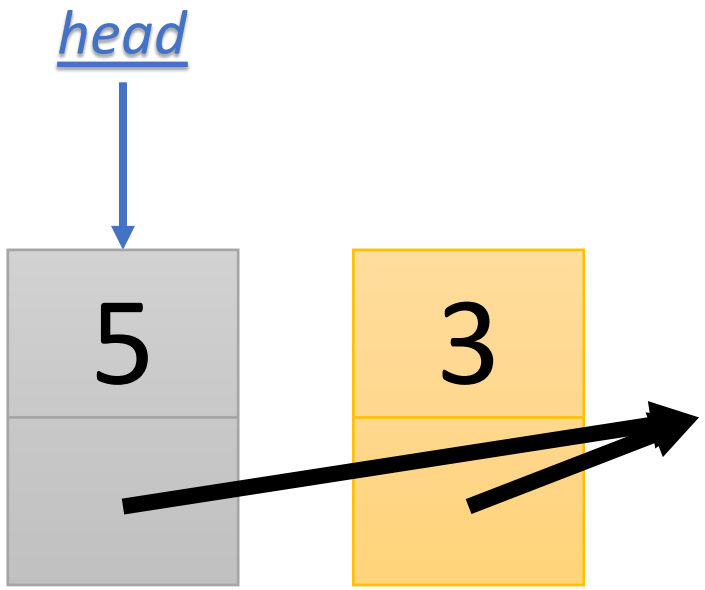
Removal from SortedList

- Example 3: removeAt (1)



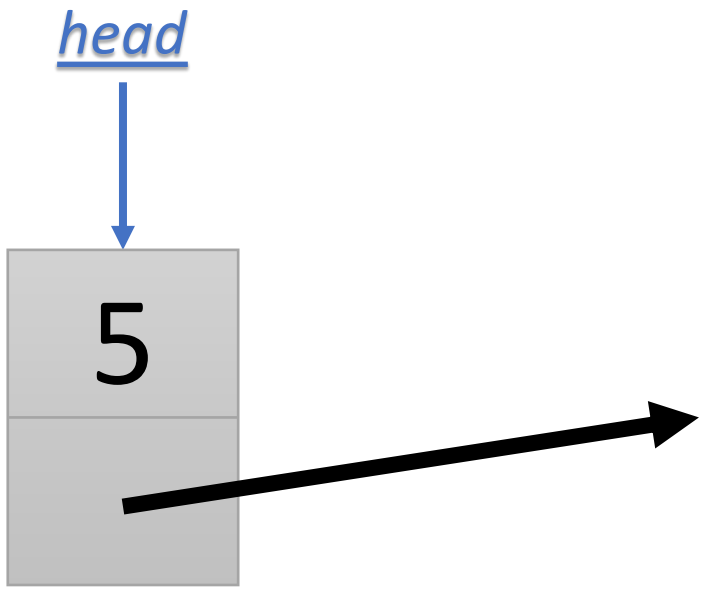
Removal from SortedList

- Example 3: removeAt (1)



Removal from SortedList

- Example 3: removeAt (1)



Removal from SortedList

- Example 3: removeAt (1)