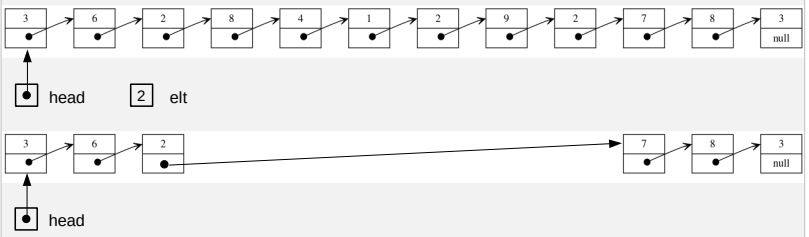
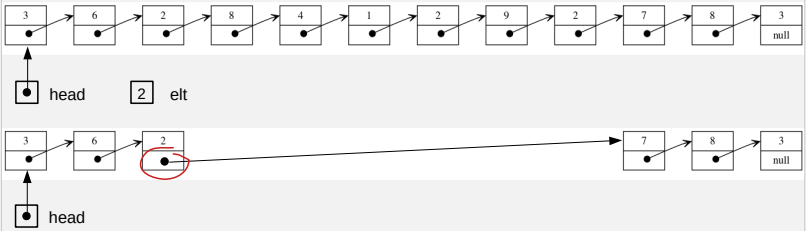
Lab 5 – Compact

- draw before and after pictures
  - task is to remove the nodes between the first occurrence of `elt` up to and including the last occurrence of `elt`



Lab 5 – Compact

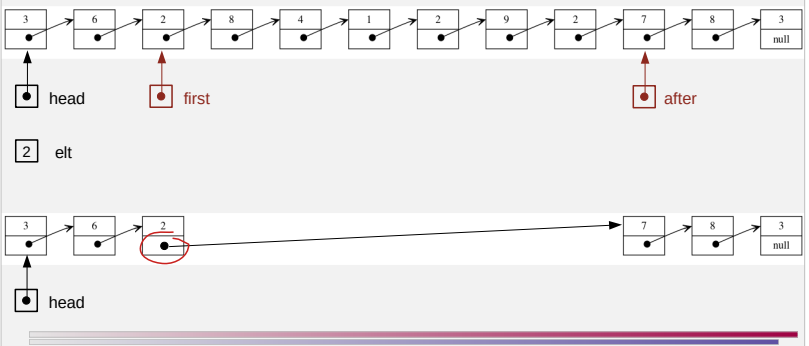
- identify what is different
  - remove nodes by relinking without them – no explicit deallocation is needed



only a single setNext is needed to do the actual compaction!

Lab 5 – Compact

- identify the convenient fingers
  - only a single setNext is needed – but we need to know the node to call setNext on and the value to setNext to
    - the “convenient fingers” point to those nodes….



Lab 5 – Compact

- implement the operation – three separate steps

```
// get first pointing to the first occurrence of elt
for ( ListNode first = head ; first != null ; first = first.getNext() ) {
  if ( first.getElt() == elt ) { break; }
}
// get after pointing to the node after the last occurrence of elt
// relink
```

  - avoid nesting later steps inside loops – this leads to needlessly complex code

```
for ( ListNode first = head ; first != null ; first = first.getNext() ) {
  if ( first.getElt() == elt ) {
    // get after pointing to the node after the last occurrence of elt
  }
}
```

Lab 5

- pattern for moving through a linked list
  - typically `runner` starts at the head and repeatedly moves to the next node…

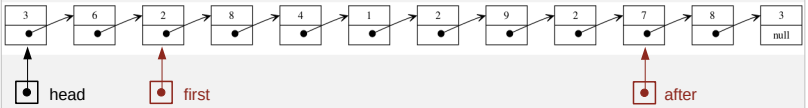
```
for ( ListNode runner = head ; runner != null ;
      runner = runner.getNext() ) {
  …
}
```

"keep going" condition may vary

  ...but starting at the head is only because that's often the only pointer to a node that we have for the list
  - also, the name `runner` is like `i` for arrays – a generic name when its only purpose is a finger keeping track of the current spot

→ adapt the pattern (variable name, starting point, and loop condition) to your specific needs

Lab 5




```
for ( ListNode after = first.getNext() ; after != null ;
      after = after.getNext() ) {
  …
}
```

- `after` is the variable we want to work with
- no need to start at the beginning (`head`) since the first node after the last occurrence for `elt` won't be before the first occurrence (`first`)