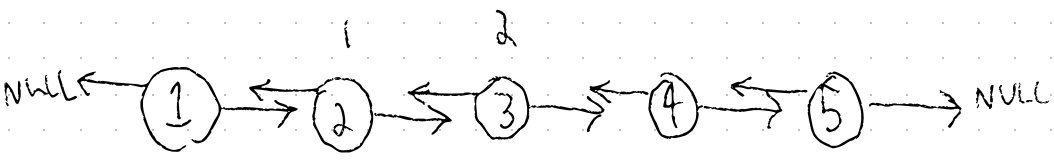
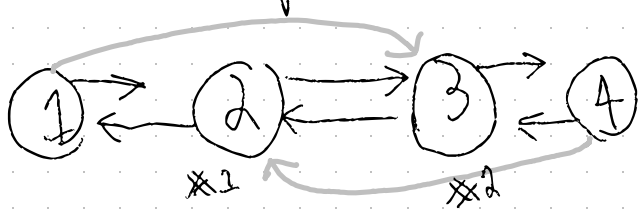
A-2, 3

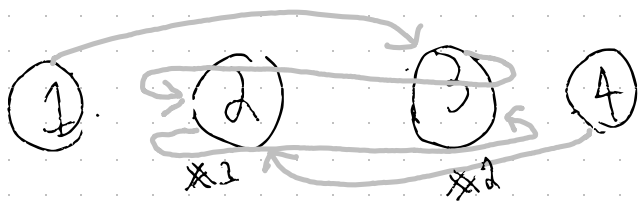
For A, I implemented a doubly linked list, and then wrote a method to swap any node with any other node. The explanation here, however, will focus specifically on the case where we are swapping a given node with its successor.



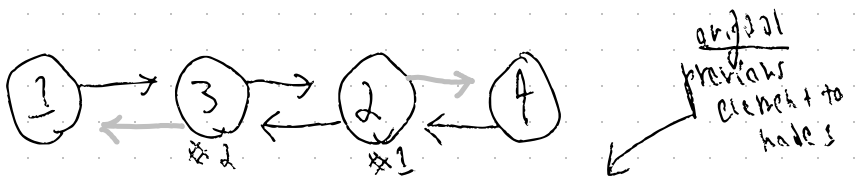
let's swap 2 and 3




```
node2Next.setPrev(node 1);
node1Next.setNext(node2);
```




```
node2.setNext(node 1);
node1.setPrev(node2);
```




```
node2.setPrev(node1Prev);
node1.setNext(node2Next);
```

for additional examples see the JUnit tests.

since the swap is just a fixed number of reference changes, once you have the node it is O(1).

However, finding the node in order to swap it is O(n) since you need to to traverse the linked list node-by-node.

This means that the overall time complexity is O(n), or O(1) if you call the overloaded method with two nodes.