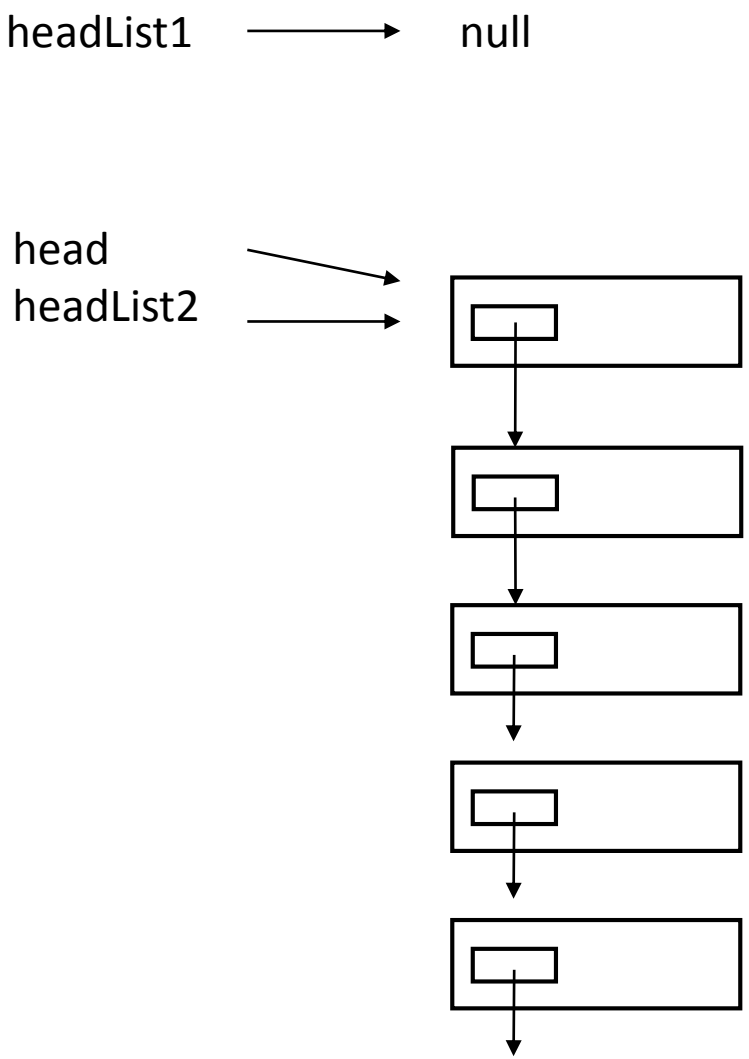
The code starts off like this:

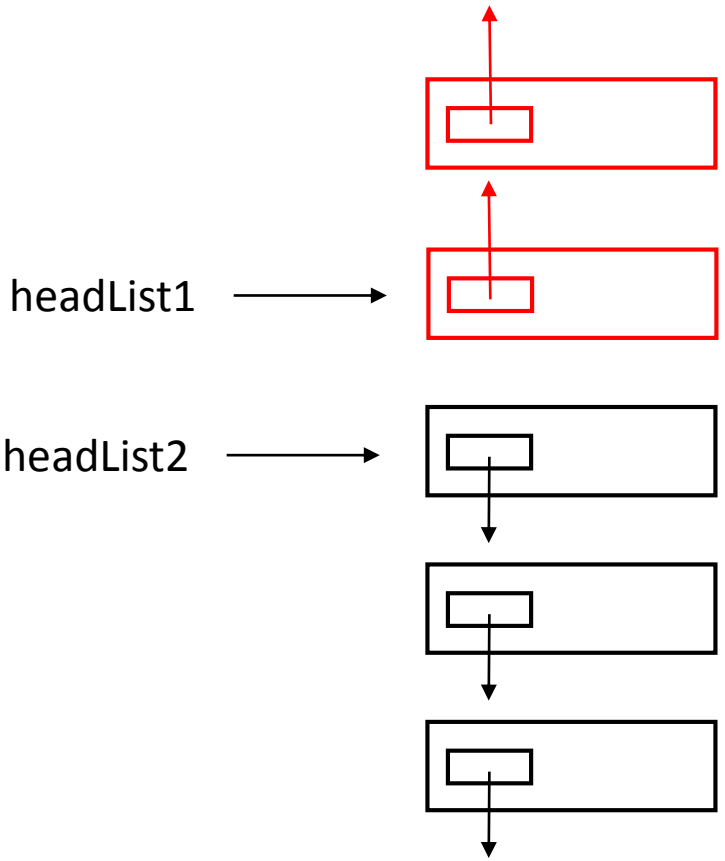
```
if (head != null){
        headList1 = null;
        headList2 = head;
```

That's just saying that if the list is not empty, then there is something to do (namely reverse the list). In that case, we're going to initialize headList1 and headList2 as above.

That gives us the situation shown on the next slide.

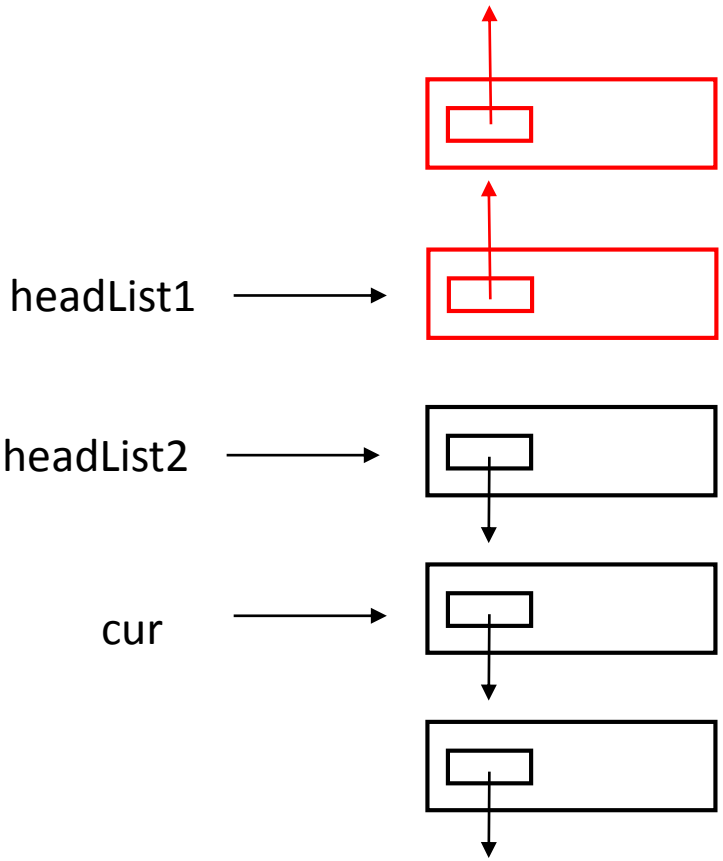

headList1
null
head
headList2

Suppose we have the general situation below in which the list is partly reversed. headList1 is the head of the part that has been reversed and headList2 is the head of the part that hasn't yet been reversed. (At the situation in the previous slide, none of the list has been reversed yet, and so headList1 points to null. In this case you think of the red nodes as not being there.)

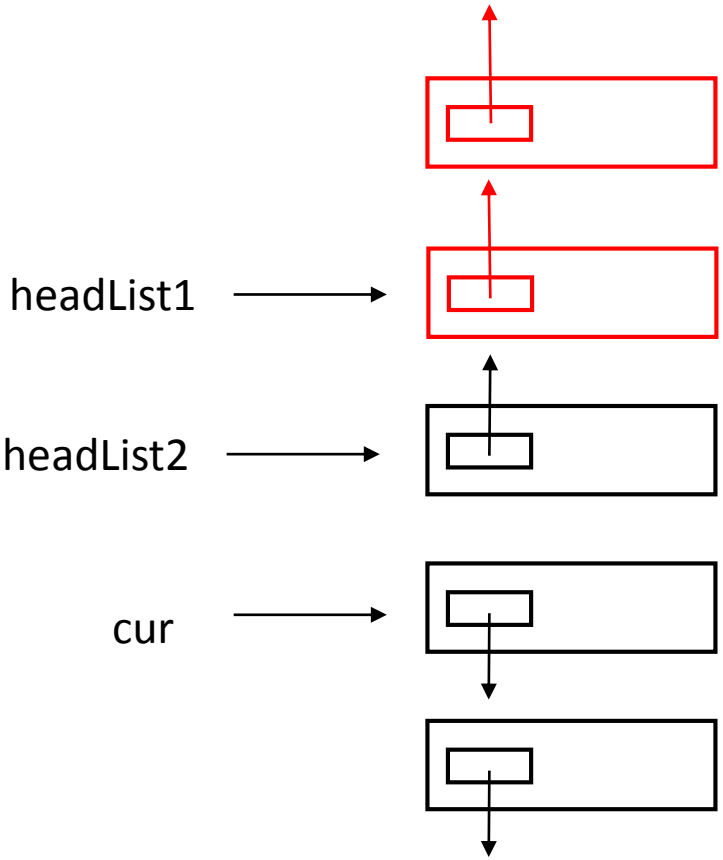


Then, supposed we run the following four instructions. I claim that this will transfer one node from list2 to list1.

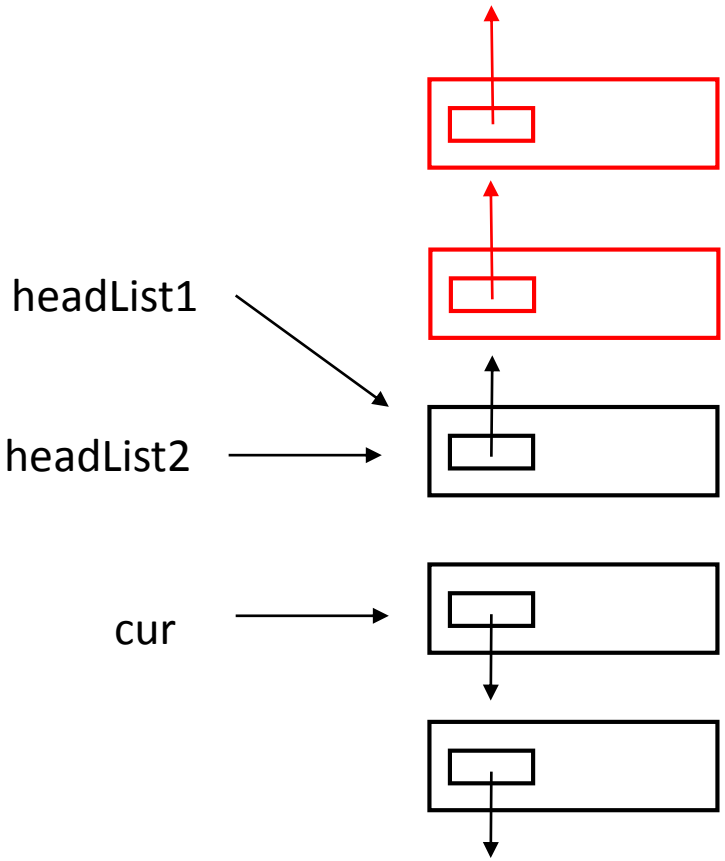
```
cur = headList2.next;
headList2.next = headList1;
headList1 = headList2;
headList2 = cur;
```



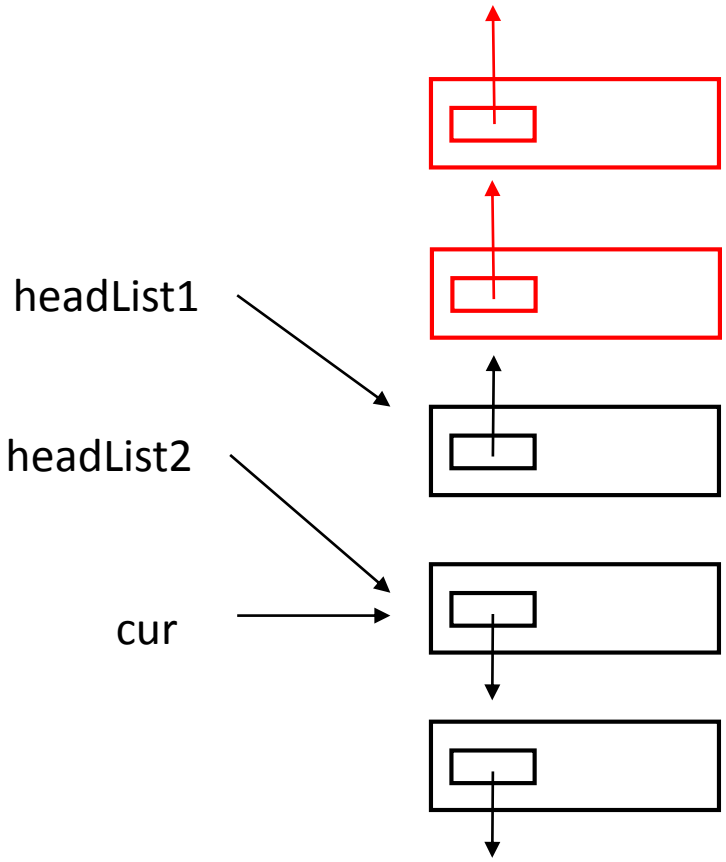
cur = headList2.next;
headList2.next = headList1;
headList1 = headList2;
headList2 = cur;




```
cur = headList2.next;
headList2.next = headList1;
headList1 = headList2;
headList2 = cur;
```




```
cur = headList2.next;
headList2.next = headList1;
headList1 = headList2;
headList2 = cur;
```




```
cur = headList2.next;
headList2.next = headList1;
headList1 = headList2;
headList2 = cur;
```

After the four instructions, we have one more node in the reversed part of the list (list1), and one fewer node in the not-yet-reversed part of the list (list2).

These four instructions are repeated over an over again in a while loop, until eventually head2List2 is null and headList1 references the last node in the original list.