Remove duplicates from LL

Problem Description: Given a sorted linked list (elements are sorted in ascending order), eliminate duplicates from the given LL, such that output LL contains only unique elements.

How to approach?

Start from the first node and compare it with the next node's data. If they are unequal or if the next node is null, move the position of the current node to the next node. If they are equal then assign the current node's next to it's next node's next. For example: let's consider the following linked list.

```
3->3->3->4->null
```

We are on the first '3' of the linked list. We'll compare this '3' with the next element which also happens to be a 3. So now, we'll assign the first '3's next to the second '3's next. This will change the linked list to:

```
3->3->4->null
```

Note!!!!: Now notice that we are still on the first '3'. So for the next iteration we won't move the current node to its next node. If we did, then we would have landed on the second '3' and then would have compared that with '4'. This would have caused us to miss the first '3' and the final output would have been the same as the LL shown just above this paragraph.

Pseudo-code for this approach is shown on the next page

```
function removeDuplicates(headNode):
  currentNode <- headNode
  while currentNode is not null:
     nextNode <- currentNode.next
     if(nextNode is null or nextNode.data != currentNode.data):
         currentNode = nextNode
     else:
         currentNode.next = nextNode.next
 return headNode
```