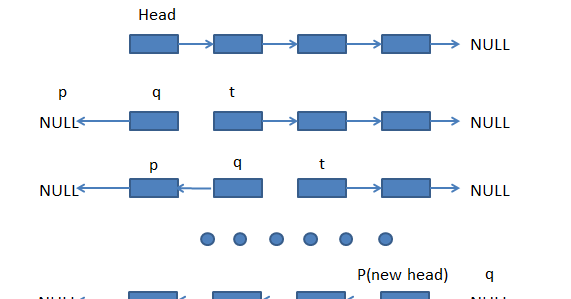
# **Reverse Linked List**

* Problem Statement

Given pointer to the head node of a linked list, the task is to reverse the linked list. We need to reverse the list by changing the links between nodes.

* Solution
* Initialize two pointers:
  + *previous as NULL*
  + *current as head*
* An important thing to note is that we initialize the previous (p) pointer as null. This is because since we are completely reversing the list, the very first node’s next pointer will now point to null, as opposed to the very last node’s next pointer pointing to null.
* While the *current* pointer is not null
  + We save *next* as a temp pointer to *current.next* because we will soon lose the ability to access the *current.next* node
  + Set *current.next* to *previous* (this is the actually act of reversing)
  + Now we move the pointers up. Previous goes to current, and current goes to next
* The while loop condition is based off of the current *c* pointer.
* We need to make sure that our current variable is assigned once to every node.
* Since we are essentially doing follow the leader, the previous *p* pointer has no way of reaching the end of the list before *c*.
* And since we can’t legally access the next pointer of a null node, we must exit the loop before trying to create a next *n* pointer.



ListNode p = null;

ListNode c = head;

while(c != null)

{

ListNode n = c.next;

c.next = p;

p = c;

c = n;

}

return p;

We return p because c will now be null after the loop terminates.