REVERSE A SINGLY LINKED LIST

☆ In this problem, we are given a singly linked list and our job is to reverse it.

For iterative answers, Brute force solution is to store all elements in an array after one pass. Then in other iteration we just modify the data. Time complexity is O(2N) and Space Complexity is O(N).

Optimal iterative solution is to reverse the links at each node in a single iteration.

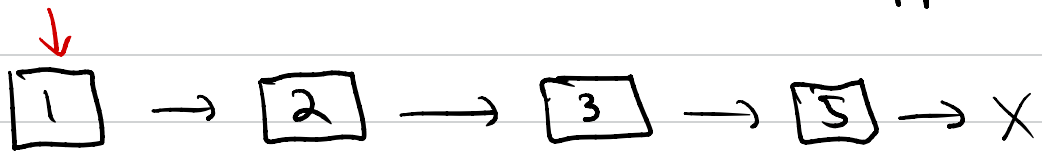
C++ :

```
Node *  reverseLL(Node * head)  {
    Node * mover = head ;
    Node * prev = nullptr ;
    Node * front = nullptr ;
    while ( mover != nullptr) {
        front = mover → next ;
        mover → next = prev ;
        prev = mover ;
        mover = front ;
    }
    return prev ;
}
```

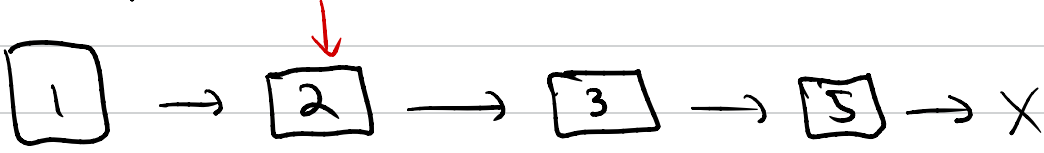
This problem also has a recursive solution. Basecase is to reverse a single element linked list.

```
Node * recursiveReverse(Node * head) {
    if(head -> next == nullptr || head
                                == nullptr) {
        return head;
    }
    Node* newHead = recursiveReverse(head ->
                                     next);
    Node * front = head -> next;
    front -> next = head;
    head -> next = nullptr;
    return newHead;
}
```

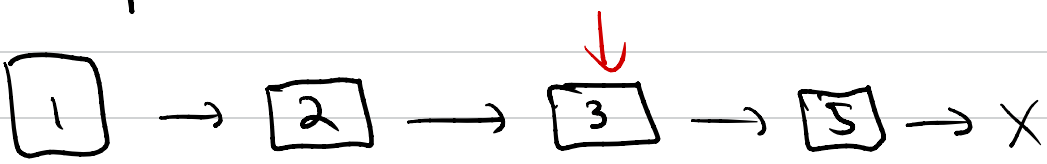
Basically, we start at the head, if it is the end of our linked list, it is returned, otherwise function is called recursively with the next element. This helps us reach the last element. When it is returned to second last element as newHead the links are swapped.



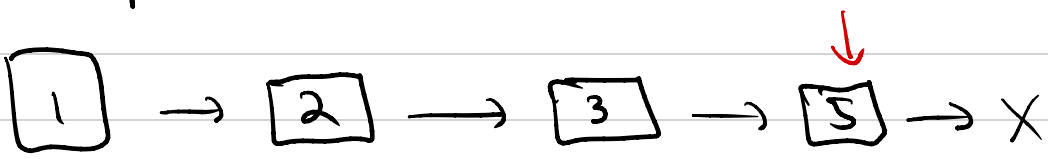
Step 1:



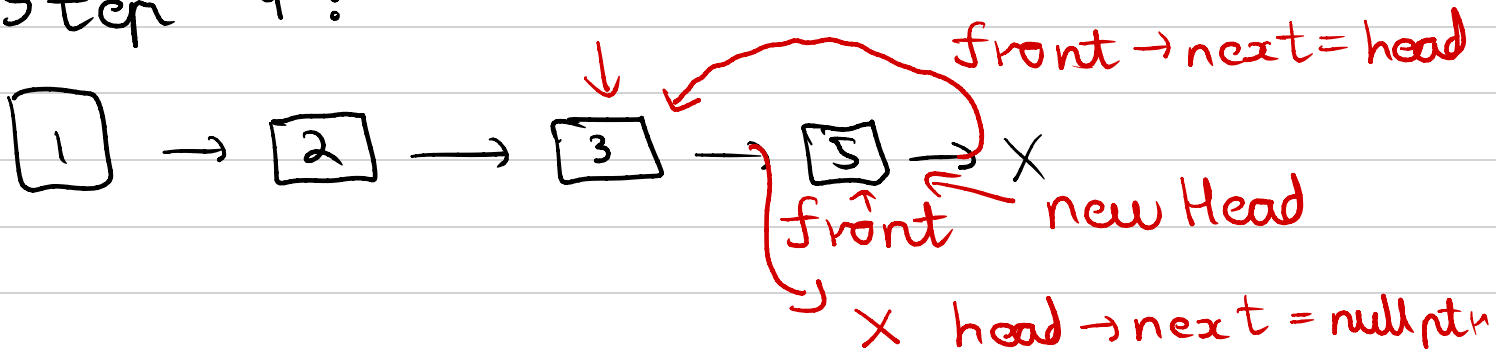
Step 2 :



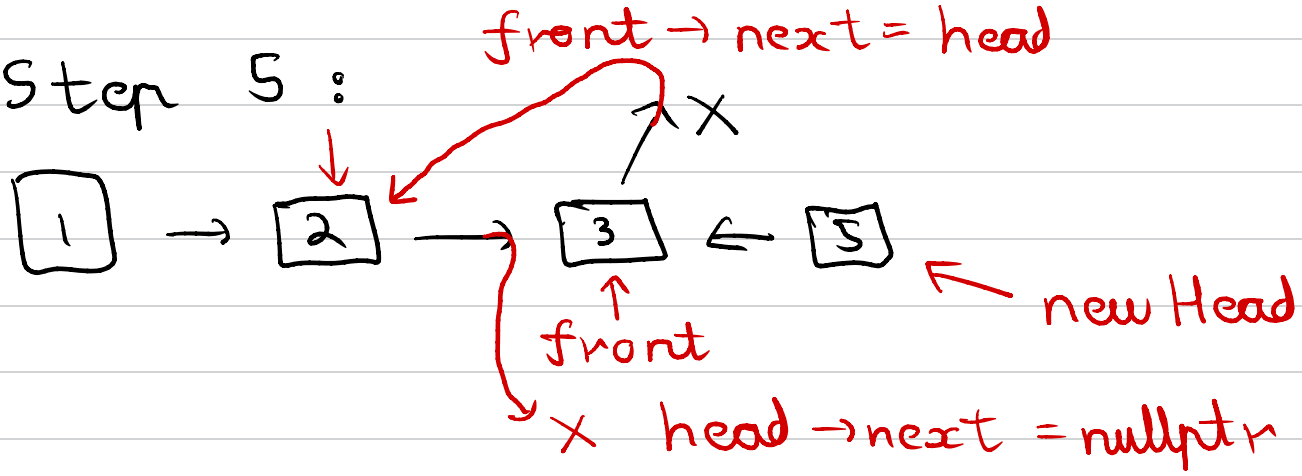
Step 3 :



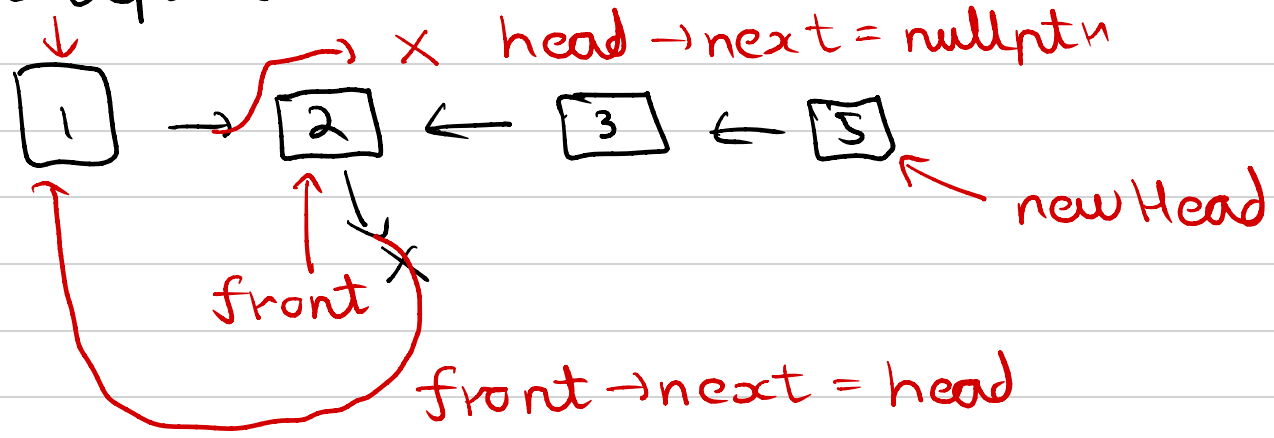
Step 4 :



Step 5 :



Step 6 :



Step 7 :