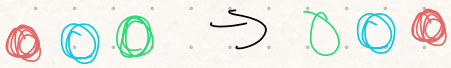
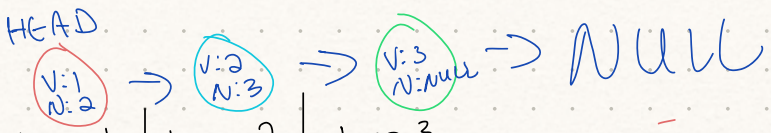
Challenge 09
- in - place reversal of linked list
- singly linked list

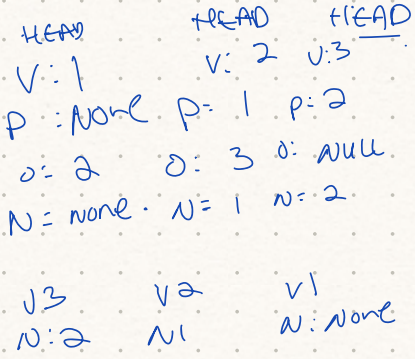
Visual:





	Before:	loop 1	loop 2	loop 3
prev:	None	1	2	3
current	Head: 1	2	3	NONE
old:	Head.next: 2	2	3	NONE
new:	None	None	1	2

WALKTHROUGH



Algorithm:
- loop through list to reverse each node order
- change the new next to the previous node
- change the old next to the current node's next
- change the previous node to the current node
- change the current node to the old next

Pseudocode:

```
prev_node = none
current_node = self.head
old_next = self.head.next
new_next = none
while old_next is not None:
    new_next = prev_node
    old-next = current_node.next
    prev_node= current_node
    current_node= old_next.
```

# palindrome
Tacocat = true
house = false.

Algorithm: P(list)
→ reverse list1, see if equal to input list
if list 2 (x) = list 1(x) reversed,
    return true
Else return false

List comprehension code draft:
if all (list 1 (x) = list 2 (x) for x in list 2):
return true
else:
return false

WALKTHROUGH of reversing list:

prev: NONE
current: HEAD (A)
old-next: HEAD.NEXT (B)
new-next: prev (NONE)

A	B	C	D
B	C	D	NONE
C	D	NONE	—
A	B	C	D

prev: NONE
current: HEAD = A
old-next: HEAD.next (B)

```
while current is not NONE:
    new-next = previous
    prev = current
    old-next = current.next
    current = old-next
```

1	C	A	C	input
2	C	A	C	reversed input
	true	true	true	

list comprehension → return true

1	C	A	B	input
2	B	A	C	reversed input
	false	true	false	

list comprehension → return false