Reverse Linked List II

Question: Reverse a linked list from position m to n. Do it in-place and in one-pass.

For example:

Given 1->2->3->4->5->NULL, m = 2 and n = 4,

Return 1->4->3->2->5->NULL.

Note: Given m, n satisfy the following condition:

$1 \leqslant m \leqslant n \leqslant$ length of list.

Solutions:

```
class ListNode(object):
    def __init__(self, x):
        self.val = x
        self.next = None


    def to_list(self):
        return [self.val] + self.next.to_list() if self.next else [self.val]


class Solution(object):
    def reverseBetween(self, head, m, n):
        """
        :type head: ListNode
        :type m: int
        :type n: int
        :rtype: ListNode
        """
        dummy = ListNode(-1)
```

```
        dummy.next = head
        node = dummy
        for __ in range(m - 1):
            node = node.next
        prev = node.next
        curr = prev.next
        for __ in range(n - m):
            next = curr.next
            curr.next = prev
            prev = curr
            curr = next
        node.next.next = curr
        node.next = prev
        return dummy.next

if __name__ == "__main__":
    n1 = ListNode(1)
    n2 = ListNode(2)
    n3 = ListNode(3)
    n4 = ListNode(4)
    n5 = ListNode(5)
    n1.next = n2
    n2.next = n3
    n3.next = n4
    n4.next = n5
    r = Solution().reverseBetween(n1, 2, 4)
    print ( r.to_list() )
```