Swap Nodes in Pairs

```
class ListNode:
     def __init__(self, val=0, next=None):
         self.val = val
         self.next = next
class Solution:
    def swapPairs(self, head: Optional[ListNode]) -> Optional[Li
        if not head or not head.next:
            return head

        dummyNode = ListNode(0)
        dummyNode.next = head
        prevNode = dummyNode
        currentNode = head
        while currentNode and currentNode.next:

            firstNode = currentNode
            secondNode = currentNode.next

            prevNode.next = secondNode #dummyNode now points to
            firstNode.next = secondNode.next #Points to the 3rd
            secondNode.next = firstNode #2nd Node now points to

            prevNode = firstNode #Pointers now move
            currentNode = firstNode.next


        return dummyNode.next
```

In this example, Dummy node is used as the starting point for the resulting list, helps handles cases where head of the list changes due to swapping

Previous Node tracks last node that has been processed

First and second node swap

Swapping Process:

- dummyNode.next points to the second node
- First node points to what was originally the third node (or a null)
- second node points to the first node

Pointer shifting:

- prevNode becomes the first node, (which is the now second in the new order) and current node moves to what was originally the third node