



# Daily Coding Problem #145

## Problem

This problem was asked by Google.

Given the head of a singly linked list, swap every two nodes and return its head.

For example, given 1 -> 2 -> 3 -> 4, return 2 -> 1 -> 4 -> 3.

## Solution

This problem can be solved either recursively or iteratively, and either by mutating next pointers or val values. The easiest implementation which yields the best space and time complexity is iteratively changing val.

In the following code, we maintain and see these properties:

- curr holds the current node that needs to be swapped with its next node.
- If curr.next is None, then there isn't anything to be swapped with, so the loop exits.
- In the loop, curr is able to jump to curr.next.next since we know curr.next exists.

```
def swap_every_two(node):  
    curr = node  
  
    while curr and curr.next:  
        curr.val, curr.next.val = curr.next.val, curr.val  
        curr = curr.next.next  
    return node
```

This takes  $O(1)$  space and  $O(n)$  time.

© Daily Coding Problem 2019

[Privacy Policy](#)

[Terms of Service](#)

[Press](#)