## 206反转链表

```go
package main

import "fmt"

type ListNode struct {
	Val int
	Next *ListNode
}

func reverseList(head *ListNode) *ListNode {
	var pre, cur, next *ListNode
	cur = head
	for cur != nil {
		next = cur.Next
		cur.Next = pre
		pre = cur
		cur = next
	}
	return pre
}

func printList(head *ListNode) {
	cur := head
	for cur != nil {
		fmt.Print(cur.Val, "->")
		cur = cur.Next
	}
	fmt.Println("nil")
}

func main() {
	l1 := &ListNode{Val: 1}
	l1.Next = &ListNode{Val: 2}
	l1.Next.Next = &ListNode{Val: 3}
	printList(l1)
	l1 = reverseList(l1)
	printList(l1)
}
```

## 103二叉树的锯齿形层序遍历

```go
package main

import "fmt"

type TreeNode struct {
	Val         int
	Left, Right *TreeNode
}

func zigzagLevelOrder(root *TreeNode) [][]int {
    var ans [][]int
    if root == nil {
        return ans
    }
    flag := true
    queue := []*TreeNode{root}
    for len(queue) > 0 {
        curLayerLen := len(queue)
        var layer []int
        for i := 0; i < curLayerLen; i++ {
            cur := queue[0]
            queue = queue[1:]
            layer = append(layer, cur.Val)
            if cur.Left != nil {
                queue = append(queue, cur.Left)
            }
            if cur.Right != nil {
                queue = append(queue, cur.Right)
            }
        }
        if !flag {
            for i, j := 0, curLayerLen-1; i < j; i, j = i+1, j-1 {
                layer[i], layer[j] = layer[j], layer[i]
            }
        }
        ans = append(ans, layer)
        flag = !flag
    }
    return ans
}

func main() {
    // 创建一个示例二叉树
    //         1
    //        / \
    //       2   3
    //      / \ / \
    //     4  5 6  7
        root := &TreeNode{
        Val: 1,
        Left: &TreeNode{
            Val: 2,
            Left: &TreeNode{
                Val: 4,
            },
            Right: &TreeNode{
                Val: 5,
            },
        },
        Right: &TreeNode{
            Val: 3,
            Left: &TreeNode{
                Val: 6,
            },
            Right: &TreeNode{
                Val: 7,
            },
        },
    }
    // root := &TreeNode{Val: 1}
    // root.Left = &TreeNode{Val: 2}
    // root.Right = &TreeNode{Val: 3}
    // root.Left.Left = &TreeNode{Val: 4}
    // root.Left.Right = &TreeNode{Val: 5}
    // root.Right.Left = &TreeNode{Val: 6}
    // root.Right.Right = &TreeNode{Val: 7}
    fmt.Println(zigzagLevelOrder(root)) // 输出: [[1], [3, 2], [4, 5, 6, 7]]
}
```