# 1-链表常用代码

链表的复制

```
// 链表的复制，添加一个头结点占位。
class Solution {
public:
    Node* copyRandomList(Node* head) {
        Node* p = head;
        Node* dum = new Node(0); // 无意义的占位头结点。
        Node* pre = dum;
        while(p != nullptr) {
            Node* node = new Node(p->val); // 复制节点
            pre->next = node;       //前节点->next = 当前
            pre = node;
            p = p->next;             // 遍历下一节点
        }
        return dum->next;
    }
};


```

经典迭代反转链表 + 递归反转链表

```
//input: ListNode* head

//普通迭代反转链表：————————————————————————————
ListNode *pre = NULL;
ListNode *cur = head;
while (cur != NULL) {
    // 反转链表四步大法
    ListNode *next = cur->next;
    cur->next = pre;
    pre = cur;
    cur = next;
}

// 递归函数，翻转链表：————————————————————————————
ListNode* reverseList(struct ListNode* head) {
    // 如果链表为空或者只有一个节点，则直接返回
    if (head == NULL || head->next == NULL) {
        return head;
    }
    // 递归翻转链表
    ListNode *newHead = reverseList(head->next);
    // 将当前节点的下一个节点的 next 指针指向当前节点，实现翻转
    head->next->next = head;
    // 将当前节点的 next 指针置空，以免出现循环引用
    head->next = NULL;
    // 返回新的头节点
```

```
27      return newHead;
28  }
```