1

在一个双向循环链表中，给出删除节点p的操作（伪）代码

```
void del(node * p) { //删除节点p
    p->prev->next = p->next;
    p->next->prev = p->prev;
    delete p;
}
```

2

给定一个数组，且数组中的元素单调不减。请你给出算法，原地删除重复出现的元素，且仅使用$O(1)$的额外空间。

```
int remove_same_elements(int * array, int size) { //返回新数组长度
    if (size == 0) return 0;
    int i = 0; //指向修改后数组的最后一个元素
    for(int j = 1; j < size; j++) { //遍历旧数组
        if (array[j] != array[i]) { //找到不一样的元素则增长数组
            i++;
            array[i] = array[j];
        }
    }
    return i + 1; //新数组长度为i + 1
}
```

3

请设计算法，在不修改链表中元素的情况下，判断一个带有头节点 $head$ 的单向链表 L 是否含有环；如果有环，请找出环的入口点。

```
node * find_circle(node * head) { //如果有环返回路口点，无环返回 nullptr
    if (head == nullptr || head->next == nullptr) return nullptr;

    node * fast_ptr = head, * slow_ptr = head; //快慢指针

    while(fast_ptr != nullptr && fast_ptr->next != nullptr) { //快指针每次走两步，慢指针每
        slow_ptr = slow_ptr->next;
        fast_ptr = fast_ptr->next->next;

        if (slow_ptr == fast_ptr) break; //相遇退出，有环
    }

    if (fast_ptr == nullptr || fast_ptr->next == nullptr) return nullptr; //如果快指针走到

    slow_ptr = head; //慢指针移回头指针
    while(fast_ptr != slow_ptr) { //每次走一步
        fast_ptr = fast_ptr->next;
        slow_ptr = slow_ptr->next;
    }
    return fast_ptr; //再次相遇的节点即为环的入口
}
```