作业六

本次作业只需要 1 个 main 函数.

单向链表节点的定义：

```
typedef struct n {
    int data;
    struct n * next;
}node;
```

Part I 单向链表的建立，展示和销毁

Task 1 编写函数 node * create(int data)，使用动态分配内存空间的方法创建一个链表节点，使用传入的 data 初始化它的 data，它的成员变量 next 值使用 NULL 初始化. 将其地址返回给它的调用者.

在 main 函数中使用它创建一个单向链表的头节点，其中 data 值为 0.

Task 2 编写函数 node * add(node * head, int data, int method)，在传入的头节点地址为 head 的单链表增加一个新节点，如果 method 为 1 在链表末尾新增，如果 method 为 0 在链表头节点前新增. 节点中 data 的值使用传入的 data 初始化. 将完成新增操作后的链表的头节点地址返回给调用者.

在 main 函数中使用它在 Task 1 中新建的单向链表头节点末尾新增一个节点，其 data 值自行指定；再在此单向链表头节点前新增一个节点，其 data 值为 0.

Task 3 编写函数 void print(node * head)，使用传入的链表节点地址，依次遍历链表中的节点并输出其中的 data，直至链表末尾.

在 main 函数中使用它展示在 Task 2 中新增过两个节点后的单向链表中所存的 data. 输出格式自行设定.

Task 4 编写函数 void delete(node * head)，使用传入的链表节点地址，依次遍历链表中的节点并依此释放每个节点所占用的内存空间，直至该链表在内存中所占的空间被全部释放.

本作业全部任务完成后，在 main 函数中调用它释放定义的所有单向链表在内存中所占用的空间.

Part II 单向链表的增删改查

Task 5 编写函数 node * insert(node * head, int position, int method, int data)，使用传入的链表节点地址，在链表中的第 position 个节点的前或后（method 为 1 在 position 前，method 为 0 在 position 后）插入一个 data 值为传入 data 的链表节点. 将完成插入操作后的链表的头节点地址返回给调用者.

在 main 函数中使用它在 Task 2 中新增过两个节点后的单向链表的第 1 个节点（即头节点）前，插入一个节点，data 值为 0；再在此单向链表的第 3 个节点后，插入一个节点，data 值自行指定；最后在此单向链表的第 5 个节点后，插入一个节点，data 值为-1. **每次**插入操作完成后**都要**调用一次 Task 3 中的 print 函数展示链表当前所存的所有 data.

Task 6 编写函数 node * del(node * head, int key)，使用传入的链表节点地址，依次遍历链表中的节点，并删除其中所有 data 值为 key 的节点，将完成删除操作后的链表的头节点地址返回给调用者.

在 main 函数中使用它在 Task 5 插入三个节点后的单向链表中，删去所有值为 0 的链表节点；再在此单向链表中删去一个你在前述任务中新增或插入的节点；然后在此单向链表中删去所有值为-1 的链表节点；最后调用此函数尝试删除一个不存在于链表中的 key. 每次删除操作完成后都要调用 Task 3 中的 print 函数展示链表当前所存的所有 data.

Task 7 编写函数 void change(node * head, int key, int data)，使用传入的链表节点地址，将该链表中所有 data 值为 key 的节点的 data 值改为传入的 data. 在 main 函数中使用合适的方案测试该函数的正确性.

Task 8 编写函数 int find(node * head, int key)，使用传入的链表节点地址，查询 data 值为 key 的节点在该链表中首次出现的位置并返回，对于链表中的第一个节点，其位置为 1. 如果 key 不存在，返回 -1.

在 main 函数中使用合适的方案测试该函数的正确性.

Part III 有序单向链表

Task 9 编写函数 node * inc_add(node * head, int data)，使用传入的链表节点地址，保持所有节点的 data 非递减顺序向其中增加节点，新增节点中的 data 为传入的 data.

在 main 函数中新建一个链表，使用合适的方案测试该函数的正确性.

Part IV 无序单向链表的排序（选做）

Task 10 编写函数 node * sort(node * head, int pattern)，使用传入的链表节点地址对链表进行排序，pattern 为 0 进行非递增排序，pattern 为 1 进行非递减排序.

在 main 函数中新建一个链表，链表至少包含 10 个节点，节点 data 随机生成，使用合适的方案测试该函数的正确性.