链表Coding Tips:

1.能够确定头(head)、尾(tail)、第i个节点

（最后一个node->next为空）

2. 弄清指向关系和操作顺序!

3. 留意特殊情况!

（例如: 为空的初始情况或链表只有1个节点）

4. 注意归还空间!

调试Tips

1. 对链表调试，重要的依然是断点(cout输出做辅助)

“数据为王”，把数据（content、甚至有时还有地址）打印出来可能会好点...

当牵涉Node交换（不仅是p->content），地址有时也要查看

2. 程序不是单靠“瞪大眼睛”看出来的，更重要的是针对测试用例，关键步骤的输出是否符合我们的“大脑”推理、手工计算

3. （对于复杂程序和测试用例）除非你有极强的逻辑思维和记忆力，否则请拿起笔和纸来...

调试Tips

```
struct Node
{   int content; //代表结点的数据
    Node *next; //代表后一个结点的地址
};
int main()
{   Node *head;
    head = input();
    sort(head);
    output(head);
    remove(head);
    return 0;
}
```

发现最后结果不对：

1. (Head, hand)记下不对的例子（测试用例）

2. 从后向前(或从前往后)排查每个函数的输出是否正确，以确定那个函数的问题

3. 发现sort不对，也有可能是input的问题

…

调试Tips

```
void sort(Node *h) //采用选择排序，小的往前放
{  if (h == NULL || h->next == NULL) return;
                  //从链表头开始逐步缩小链表的范围
    for (Node *p1=h; p1->next != NULL; p1 = p1->next)
    {
         Node *p_min=p1; //p_min指向最小的结点，初始化为p1
                  //从p1的下一个开始与p_min进行比较
        for (Node *p2=p1->next; p2 != NULL; p2=p2->next)
             if (p2->content < p_min->content)
                       p_min = p2;
         if (p_min != p1)
        {     int temp = p1->content;
              p1->content = p_min->content;
              p_min->content = temp;
         }
    }
}
```

当确定一个函数有问题后...

要 求

除完成教材第5章相应习题 + 周二上机习题外，

首先把PPT上的所有代码、习题自己Coding一遍！