1. TA给出的类可能是不完整的，你可能需要添加、删减或更改一些东西
2. TA给出的测试函数可能是不完整或者是有错误的，最后是自己把所有要测试的内容补充完整
3. 有关模式的题，看上去好像很纠结，其实多数时候就是在坑人。看着输出直接补充类的相关内容就好
4. 复制粘贴需谨慎
5. TA给的类的声明里面某些东西可能是错的！！
6. 不要抽风把某些符号漏写
7. 注意TA给了什么东西，在sicily上提交的时候注意不要多复制，导致重定义
8. 如果认为算法没错，在类外定义出错的时候，可以尝试把定义直接放在类里面（有些深层含义的错误吾等无法体会……）

++++++++++++++++++++++++++被遗漏或之间没记住的知识点++++++++++++++++++++++++++

1. 运算符重载中注意返回类型是否有引用（若有给出class，注意不要抄错）。+、-、\*、/等是不返回引用，而一元操作符如 += 等返回引用
2. 在函数中声明传入参数是Observer，如，而Observer只是虚基类，实际情况是，即传入参数是子类，这种情况应该使用动态绑定，即而不能是静态绑定即，否则编译器会报错：
3. 队列的实现可以用循环数组或者链表。循环数组主要就是在到达最大下标的时候可以转回最小下标就可以了。链表暂时还不会用。
4. 关于计数器，static 要在类外初始化。
5. 关于错误提示“非常量引用的初始值必须为左值”，

实际上我定义了的是是想要把一个Student的对象传引用调用。但main函数中传入的是A的地址。因此，把传入类型改为指针就可以了。

1. 静态数据成员要通过静态成员函数访问
2. 使用类模板时，在类外定义函数时要注意加上
3. delete只用删除指针类型，不能这样单独删除数组中的一个元素
4. 关于拷贝构造函数。这种是隐式拷贝构造而不是简单的赋值操作？好吧，区别在哪？

这个要看清楚了，两种方式不一样的：

如果是：

Element C；

C = A；

这样子就是调用运算符重载 = ，如果没有自己重载，就是默认的重载函数，直接把所有的成员变量全都赋值一遍。

如果是：

Element C = A；

这样子就相当于Element C(A)；

拷贝构造函数无疑了吧…

这两者的区别呢…一个是调用运算符重载，一个是调用拷贝构造函数，这两个函数的不同就是他们的区别了…

++++++++++++++++++++++++++++课堂小测相关知识点+++++++++++++++++++++++++++++++

第一题：函数重载

第二题：类的成员函数的定义；斐波那契数列：F0=0，F1=1，Fn=F(n-1)+F(n-2)（n>=2，n∈N\*）

第三题：类模板；排序（可以直接用sort(数组名,数组末地址），默认字典序升序排列，需要头文件#include<algorithm>)

第四题：无明显知识点。可以尝试开向量

第五题：级联，要返回引用。

第六题：完成复数类。运算符重载。

第七题：虚继承，多重继承时为了避免歧义；

第八题：函数模板；排序（同直接调用sort）

第九题：函数模板；函数重载；

第十题：计算用深搜；全排列用递归；

++++++++++++++++++++++++++++曾经出现的问题+++++++++++++++++++++++++++++

第三题：在函数里这样，编译器会报错：原因大概是因为len在这里只是形参，没有具体的大小，但是数组初始化的时候必须给定大小。

WA原因：输出格式看错……

没有特别标明的排序都是按字典序么？？

第四题：按照第一次直接在原来的字符串上判断的算法，会出现这种错误。修改后，把括号部分先提出来，每找到一对合格的（）就把这一对（）删掉，错误被修正。

第六题：错误原因为print()里输出没有输出换行符，一直WA

第七题：需要注意继承的顺序，与的继承顺序不一样，构造函数的调用顺序不一样。

与输出相同，，前者是有一个基类指针指向派生类对象？这样写的意义何在？

第九题：指针型的和非指针型的要分开。曾经尝试合在一起，但因为指针要先从地址找到数值而后者是直接比较，没办法做到。

第十题：算法来自百度，修改了0为第一个数字时运行出错的问题（中间一个if的出栈放错了地方）；

全排列（算法同样来自百度）同时判断是否为24点时，一开始是每排列一次就判断然后输出，最后发现输出了24次“YES”或“NO”；后来设定一个bool变量pd初始化为false，把全排列输出改成了如果是24点就把pd的状态改为true；但是因为一开始不是传地址，所以不管怎样输出的都是false；

到现在都不知道为什么WA

【反正最后那个苦逼的算法是测试用例全过但sicily上WA了】

+++++++++++++++++++++++++++++++++++++++++++++++++++++