C++程序设计作业2

姓名：\_\_\_\_\_\_\_\_ 学号：\_\_\_\_\_\_\_\_ 课程编号：\_\_\_\_\_\_\_\_

1. **选择题**
2. （不定项选择题）C++中关于运算符重载说法错误的是（ ）
   1. 在重载二元运算符时（如+或者-），两个操作数的类型要保持一致  
      不一定
   2. 下标运算符[]可以用成员函数的方式重载，也可以用外部函数（友元函数）的方式重载  
      有些[双目运算符](https://so.csdn.net/so/search?q=%E5%8F%8C%E7%9B%AE%E8%BF%90%E7%AE%97%E7%AC%A6&spm=1001.2101.3001.7020)是不能重载为友元函数的，比如赋值运算符=、函数调用运算符()、下标运算符[]、指针运算符->等
   3. 流操作运算符（<</>>）可以用成员函数的方式重载  
      C++流操作符只能使用非成员函数方式重载
   4. 当用外部函数模式重载运算符时，friend友元声明不是必须的  
      在某些场景下需要从外部访问类的私有/保护成员，这时我们需要用到友元（关键字friend）
3. （不定项选择题）下列关于自定义类型转换描述正确的是（ ）
   1. 自定义类型的隐式类型转换可以通过重载类型转换操作符和构造函数实现  
      系统类型 -> 系统类型：系统内置转换

内置类型 -> 自定义类型：构造函数

自定义类型 -> 内置类型：重载类型转换操作符

自定义类型 -> 自定义类型：构造函数/重载类型转换操作符

* 1. 重载类型转换操作符能够把其他类型转换为当前类型
  2. 在构造函数前添加explicit关键字能显式声明当前构造函数参与隐式类型转换  
     除非设计时需要构造函数参与隐式类型转换，否则接受单个参数的构造函数用explicit修饰，避免错误触发转换  
     意味着用explicit修饰的构造函数不参与隐式类型转换
  3. 当函数参数类型为指针或者右值引用时，不允许发生隐式类型转换  
     当函数参数为指针或者左值引用时，是不允许隐式类型转换的

1. （不定项选择题）关于自定义类型的特殊构造函数，下面说法正确的是（ ）
   1. 拷贝构造函数的参数必须是const引用类型
   2. 自定义拷贝构造函数加入深拷贝逻辑后，即可实现自定义类型的赋值深拷贝操作
   3. 改变构造函数的初始值列表顺序，就能够改变类中成员变量的初始化顺序
   4. 移动构造函数需要接管传入右值对象内指针指向内容，并确保传入对象能正确析构  
      移动构造函数不分配任何内存，只是简单的资源移动而已
2. （不定项选择题）关于左值和右值下列描述正确的是（ ）
   1. std::move能够把左值移动为右值，移动之后的源左值对象不可继续使用，只能等待析构操作  
      虽然将资源给了右值对象，但是左值对象并没有立刻析构，只有在左值对象离开了自己的作用域的时候才会析构
   2. void Add(const int& val)函数，调用int a = 1; Add(a + 1);是正确的使用方式  
      有const正确，a+1为右值
   3. int&& rb = int(0) + 1; 是正确的右值绑定方式  
      正确，int（0）+1为右值
   4. std::move函数左值转右值时会调用一次拷贝构造  
      优先调用移动构造函数
3. （不定项选择题）关于函数的内存模型，下列说法正确的是（ ）
   1. 函数中的static变量会保存在函数的栈中  
      在静态存储区中，堆栈是动态存储区中
   2. 不要在函数内部定义过大的局部数组，可能会耗尽函数的栈空间  
      局部数组变量用栈存储，栈的容量远远小于堆的容量，因此，容易出现段溢出或越界错误
   3. 函数的返回值可能通过生成临时变量返回，也可能通过寄存器直接返回  
      返回值为值类型时。使用临时对象（temporary object）来保存函数的返回值。
   4. C++编译器通过RVO等机制可能优化掉函数返回值的拷贝操作
4. （不定项选择题）关于派生和继承，下面说法正确的是（ ）
   1. 当基类包含static成员变量时，派生类会和基类共享同一个static全局成员
   2. 派生类构造时，通过初始值列表调用基类的构造函数，我们可以通过调整初始值列表的顺序，先初始化派生类自身的成员，再初始化基类  
      不能
   3. 基类类型的指针一定能够指向派生类的对象  
      基类类型的指针只能访问基类的成员变量，并且只能调用基类的成员函数
   4. 派生类析构时先调用基类的析构函数，再调用派生类的析构函数  
      派生对象析构时先调用派生类的析构函数，再自动调用基类的析构函数，无需人工干预
5. （不定项选择题）关于函数覆盖和重载，下面说法正确的是（ ）
   1. 如果基类有成员函数void Add(int i)；我们可以在派生类中定义新函数void Add(float i)；从而和基类同名函数实现重载功能  
      是盖不是重载
   2. 只有函数签名相同的情况下，派生类的函数才会覆盖基类的函数  
      派生类的同名函数（非同签名）也会隐藏基类函数
   3. 当基类成员函数被派生类覆盖后，可以用指向基类的指针调用基类被覆盖的函数
   4. 把基类对象地址赋值给派生类的指针，该指针只能调用基类被覆盖的成员  
      基类类型的指针只能访问基类的成员变量，并且只能调用基类的成员函数
6. （不定项选择题）下面关于多态说法正确的是（ ）
   1. 虚函数不能为inline函数  
      内联函数不能作为虚函数
   2. 虚函数是C++实现静态多态性的主要方式  
      ？
   3. 虚函数在派生类中被重写时也必须用virtual关键字声明  
      如果派生类不重写虚函数，则虚函数表直接使用父类的虚函数地址
   4. 虚函数表中不包含类中其他非虚函数的地址  
      非虚函数调用不经过虚函数表
7. （不定项选择题）下面关于虚函数说法正确的是（ ）
   1. 构造函数和析构函数都可以定义为虚函数  
      构造函数不能定义为虚函数  
      析构函数可以定义为虚函数
   2. 当派生类存在动态申请内存时，基类析构函数必须生命为虚函数  
      当在派生类动态申请内存时，基类的析构函数必须定义为虚函数
   3. 虚基类的含义是类型的所有成员函数都为纯虚函数  
      包含纯虚函数的类被称为虚基类（Abstruct Class），不一定所有函数都是纯虚函数
   4. 和静态绑定相比，动态绑定可能会影响系统的性能
8. （不定项选择题）下面说法正确的是（ ）
   1. delete []和delete在释放C++内置类型数组时效果是一样的  
      不一样（参考上节课作业）
   2. C语言malloc和C++的new关键字作用完全相同  
      申请内存位置不同
   3. sizeof计算一个对象大小时，不包含static成员  
      sizeof计算类的大小时，只计算（非static的）数据成员变量即可，不用考虑那些成员函数
   4. sizeof可以用来获得在堆上动态申请的数组长度  
      sizeof不能求得动态分配的内存的大小

**二、编程题**

动态内存的管理很容易出现问题：分配动态内存忘记释放会造成内存泄漏、尚有指针指向内存时就释放会造成非法内存引用、重复释放内存会造成程序崩溃……

为了更容易也更加安全地使用动态内存，C++11后标准库提供了智能指针类型。阅读下述资料，实现管理动态int对象的简化版智能int指针类shared\_int和unique\_int。

参考资料：

(1) 智能int指针对象默认构造存储空指针。此外提供int值（实际智能指针值构造函数的版本略有不同，此处为简化问题）和int指针版本的构造函数。注意，需禁止int指针向智能int指针的隐式转化。

(2) 智能int指针对象作为内置int指针的“高级”版，兼容内置int指针的判空、解引用操作。

(3) unique\_int对象独占所指向的对象，分配动态内存后仅支持转移对象所有权，不支持任何拷贝行为。

(4) shared\_int对象支持拷贝，但拷贝时不会进行深拷贝，而是令所有拷贝对象共享相同的内存地址。利用(5)中描述的引用计数技术，当对象的引用个数为0时，自动释放分配的动态内存。

(5) 引用计数：

a. 除初始化对象外，构造函数(不含拷贝/移动构造函数)还额外创建一个计数器，记录有多少对象与当前对象共享内存。构造时仅有一个对象，因此计数器初始化为1；

b. 拷贝构造函数不分配新的内存、不分配计数器，而是拷贝动态内存地址与计数器。每进行一次拷贝过程，计数器增加1；

c. 析构函数递减计数器值，当计数器值变为0时，释放共享内存状态；

d. 拷贝赋值运算符同时影响原内存计数器和新内存计数器，同c理，一旦某内存计数器减至0，释放该内存。

要求：

1. 实现满足上述资料描述的shared\_int类，以及unique\_int类。可看成两道小题，分别将两个类实现在两组程序代码中，需提供充分验证程序验证相关功能。

2. 合理实现shared\_int与unique\_int类的特殊函数（构造、析构、拷贝控制函数），达到共享&自动释放（shared\_ptr）/ 独占（unique\_ptr）的效果。

3. 使用智能指针对象时，和内置指针的用法应相同（支持解引用、支持判断指针是否为空等）

4. 附加功能实现：实现智能int类的移动语义函数、swap函数

5. 多线程竞争等问题可暂不考虑。

作业提交方式：

选择题答案写在单独的文本文件中，格式如：

1. ABCD
2. ABCD
3. ABCD
4. ……

编程题要包含可正确编译执行的源代码（包含shared\_int与unique\_int对象的实现和相应的测试程序）以及可以执行编译的脚本build.sh

11月25日24:00之前讲包含作业答案、程序源代码的压缩包以姓名-学号-作业2的格式提交到邮箱csy1062022@163.com

有特殊情况需要迟交作业的要和助教以及教师申请。