北 京 交 通 大 学 考 试 试 题（A）

课程名称：C++程序设计　学年学期：2022—2023 学年第 1 学期

课程编号：M410010B　开课学院：软件学院

出题教师：黄博阳、闫晗、李宗辉

学生姓名：_____　学号：_____　任课教师：_____

学生学院：_____　班级：_____

题　号	一	二	三	四	总分
得　分					
阅卷人					

请将第一、二题答案，写在 1 号答题纸上，拍照后的文件，命名为 1.jpg

1 号答题纸　姓名：　学号：

一、判断题

1、_____ 2、_____ 3、_____ 4、_____ 5、_____

6、_____ 7、_____ 8、_____ 9、_____ 10、_____

二、选择题

1、_____ 2、_____ 3、_____ 4、_____ 5、_____

6、_____ 7、_____ 8、_____ 9、_____ 10、_____

一、判断题（10 分，每题 1 分）：

1. C++语言设计时，力求对用户自定义类型与内置类型有相同或相近的语法支持，运算符重载等语法体现了这样的 C++哲学。（✓）

2. C++提供了一些语法对 C 语言中的写法进行替换，从而达到更安全的特点。例如 C++11 后建议使用 nullptr 替换 C 语言风格的 NULL，使用 const 修饰的常量替换宏变量，使用 inline 函数替换宏函数。（✓）

3. 尽管 C++兼容 C 语言，但通过 C 语言函数 malloc 分配的空间是不能通过 C++中的 delete 函数进行释放的。（✗）

4. 以下代码定义了长度为 5 的 int 数组，并将每个元素初始化为 0。（✗）

```
const int len = 5;
int arr[len];
```

5. 下述代码出现的变量 a、变量 b 以及 b 所指向的内存分别是静态存储区、栈区和（堆）动态内存区。（✓）

```
void fun() {
    static int a = 2022;
    int* b = new int[2048];
    // 其他运算逻辑
    delete[] b;
}
```

6. 以下代码打印输出的是"cc hh"。（✗）

```
char arr[5] = {0};
arr[0] = 'c';
arr[1] = 'c';
arr[3] = 'h';
arr[4] = 'h';
std::cout << arr << std::endl;
```

7. 下列代码通过虚继承的方式保证对象 D 中只有一份基类 A 的实例。（✓）

```
class A {…};
class B : public A {…};
class C : public A {…};
class D : virtual public B, virtual public C {…};
```

8. 以下模板函数 Add 可以被正确调用。（✗）

```
template<typename T>
T Add(T a, T b) {
    return a + b;
```

```
}
int main(){
    int a,b=3;
    double c=1.0;
    a=Add(b,c);
    return 0;
}
```

9. 使用 ofstream 对象进行文件写入操作时，打开的文件必须已经存在，ofstream 对象打开不存在的文件会让程序崩溃。（ ）

10. std::array 是 C++标准模板库的重要组成部分，与 std::vector 不同的是创建长度为 n 的 array 对象时，n 必需是常量。（ ）

二、选择题（20 分，每题 2 分）：

1. const 关键字修饰指针时，既可修饰指针变量，也可修饰指针指向的对象，以下说法错误的是（ ）

```
int x = 1;
```

A、const int *p1 = &x; // p1 表示指向 const 数据的指针
B、int* const p2 = &x; // p2 表示 const 指针
C、int const *p3 = &x; // p3 表示 const 指针
D、const int* const p4 = &x; // p4 表示指针和所指的内容都是 const

2. 在 C++的继承关系中，基类的私有成员不能在子类的函数成员中直接访问的是（ ）
A、公有继承（Public） B、保护继承(Proctected)
C、私有继承（Private） D、选项 A、B、C 均正确

3. 下列关于模板描述正确的是（ ）
A、模板函数隐式实例化调用时，函数形参可能出现隐式类型转换
B、模板函数的实例化是在程序运行过程中进行的
C、与正常函数相同，模板函数的执行逻辑也不应该定义在头文件中，否则链接时可能出现符号冲突
D、对同名函数的函数重载和模板可以同时使用

4. 关于继承和派生下列说法错误的是（ ）
A、私有继承时基类保护成员在派生类中会变成私有访问属性，所以派生类无法访问基类的保护成员
B、当基类中任一函数为纯虚函数时，该基类被称为抽象类，抽象类不能被实例化
C、私有继承时无法隐式地把基类类型的指针指向派生类（即 Implicit upcasting）
D、虚继承可以用来保证多继承时公共基类对象的唯一性

5. 关于虚函数和多态下列说法正确的是（ ）

A、公有继承、私有继承和保护继承都可以将基类函数声明为虚函数实现多态性

B、通过将派生类的析构函数声明为虚函数，可以实现 delete 释放指向派生类的基类指针时自动调用派生类的析构函数

C、构造函数不应该被声明为虚函数，而析构函数则应声明为虚函数

D、只有被派生类重写的虚函数才会出现在虚函数表中

6. 关于类下列说法错误的是（ ）

A、当开发者定义类的带参数构造函数时，编译器不会自动生成不带参数的默认构造函数

B、构造函数的返回值类型为该类型本身

C、拷贝构造函数的参数应使用引用传递而不是值传递

D、假设 T 为自定义类型，T t1; T t2 = t1; 在定义 t2 时调用了拷贝构造函数

7. 关于 C++函数下列说法正确的是（ ）

A、inline 函数在运行时无需进行函数调用

B、两个函数名和参数列表均的相同函数，但返回值类型不一样，他们是函数重载

C、void func(int a = 1, float b = 2.0, double c); 是正确的函数声明

D、void f2unc(int a, float b, double c); 是正确的函数声明

8. 下列说法错误的是（ ）

A、一个 C++可执行文件中只能存在一个 main 函数

B、当使用 delete 删除自定义类型对象时，会自动调用对象的析构函数

C、当派生类的对象被释放时，先调用基类的析构函数然后调用派生类的析构函数

D、C++的抽象类中也可以定义构造函数和析构函数

9. 类 Person 是基类，Worker 公有继承了 Person 且定义了虚函数 printJob(),以下表达错误的是（ ）

A、Worker worker; Person* person=&worker;

B、Person person; Worker* worker=&person;

C、Person& person=Worker(); person.printJob();

D、Worker worker; worker.printJob();

10. 类 Complex 是一个复数类，为了支持复数的加法运算，以下说法正确的是（ ）

A、在类 Complex 中重载运算符加：Complex operator+(Complex& a, Complex& b);

B、在类 Complex 中重载运算符加：Complex operator+(Complex a, Complex b);

C、在类 Complex 外重载运算符加：Complex operator+(Complex& a, Complex& b);

D、在类 Complex 外重载运算符加：Complex operator+(Complex& a);

请将第三题答案，写在 2 号答题纸上。拍照后的文件，命名为 2.jpg

2 号答题纸　　姓名：　　学号：

三、程序解答题

1、

2、

3、

4、

5、

6、

7、

8、

9、

10、

三、程序解答题（40 分，每题 4 分）：

1. 某同学实现如下函数，判断并返回两个 string 较短的 string

```
std::string shorterString(std::string s1, std::string s2) {
    std::string s = s1.size() < s2.size() ? s1 : s2;
    return s;
}
```

学习通过引用传递参数能减少不必要的拷贝操作后，该同学意识到上面的代码发生多次不必要的拷贝，为优化性能，调整为如下代码：

```
std::string& shorterString(const std::string& s1, const std::string& s2)
{
    std::string s = s1.size() < s2.size() ? s1 : s2;
    return s;
}
int main(){
   std::string s1,s2;
   …
   std::string& a = shorterString(s1,s2);
   …
   return 0;
}
```

请问：改动前调用函数 shorterString 进行了多少次字符串的拷贝（不考虑返回值带来的拷贝）？改动后的代码可以达成他的目标吗？如果能，请计算改动后字符串拷贝的次数（不考虑返回值带来的拷贝）；如果不能，应该如何改动才可以，请写出修改后的代码。

2. C++类通过访问限定符实现面向对象的封装特性，请阅读以下程序：

```
#include <string>
class Base {
private:
    void privateBaseFun() {}
    int privateBaseInt_;
protected:
    void protectedBaseFun() {}
    float protectedBaseFloat_;
public:
    void publicBaseFun() {}
    std::string publicBaseStr_;
};
class Derived : public Base {
public:
    void testInner() {
        int innerInt = privateBaseInt_;          // 1
        float innerFloat = protectedBaseFloat_;  // 2
        std::string innerStr = publicBaseStr_;   // 3
        privateBaseFun();                        // 4
        protectedBaseFun();                      // 5
```

```
        publicBaseFun();                                    // 6
    }
};
int main() {
    Derived derived;                                        // 7
    int outerInt = derived.privateBaseInt_;                 // 8
    float outerFloat = derived.protectedBaseFloat_;         // 9
    std::string outerStr = derived.publicBaseStr_;          // 10
    derived.privateBaseFun();                               // 11
    derived.protectedBaseFun();                             // 12
    derived.publicBaseFun();                                // 13
    return 0;
}
```

试回答上述代码编译阶段会报错的行号（1 到 13）是哪些？

3. 请阅读以下程序后，回答问题：

```
class A {
private:
    static int v;
    static int s;
    int i;
public:
   A(int i){
       this->i=i;
       this->s=1;
       std:cout << s + i*v++ << std::endl;
   }
}
int A::v=0;
int A::s=0;
int main(){
    A a(1);
    A b(2);
    A c(3);
    A d(4);
    return 0;
}
```

请问：上述程序输出的结果是什么？

4. 请阅读以下程序后，回答问题：

```
class A {
public:
     A(){};
     A(const A& a){std::cout<< “A” <<std::endl;};
};
class B {
     B(){};
     B(const B& b){std::cout<< “B” <<std::endl;};
};
class C {
     C(){};
     C(const C& c){std::cout<< “C” <<std::endl;};
};
class D : public A {
private:
    B b;
    C c;
public:
    D（C& c, B& b, A& a）:c(c),b(b),A(a){std::cout<< “D” <<std::endl;};
};
int main(){
    A a;
    B b;
    C c;
    D d(c,b,a);
    return 0;
}
```

请问：执行上述程序后的输出是什么？

5. 请阅读以下程序后，回答问题。

```
class A {
public:
    A operator+(int i) {
        std::cout << "A" << std::endl;
```

```
        return *this;
    }
};
A operator+(int i, const A a) {
    std::cout << "B" << std::endl;
    return a;
}
int main(){
    A a;
    A b=1+(a+2);
    return 0;
}
```

请问：上述代码的输出结果是什么？试解释原因。

6. 请写出下面代码输出结果：

```
#include <iostream>
class A {
public:
    A() { std::cout << "A()" << std::endl; }
    ~A() { std::cout << "~A()" << std::endl; }
    virtual void fun() { std::cout << "A.fun()" << std::endl; }
};
class B : virtual public A {
public:
    B() { std::cout << "B()" << std::endl; }
    ~B() { std::cout << "~B()" << std::endl; }
    void fun() { std::cout << "B.fun()" << std::endl; }
};
int main() {
   A* p = new B;
   p->fun();
   delete p;
   return 0;
}
```

7. 下列场景中，能够形成重载的函数有 1 3

①✓

```
void fun1(int a) { std::cout << “void fun1(int a)” << std::endl; }
void fun1(float a) { std::cout << “void fun1(float a)” << std::endl; }
```

②

```
void fun2(int a) { std::cout << “void fun2(int a)” << std::endl; }
void fun2(const int a) { std::cout << “void fun2(const int a)” << std::endl; }
```

③

```
void fun3(int* a) { std::cout << “void fun3(int* a)” << std::endl; }
void fun3(const int* a) { std::cout << “void fun3(const int* a)” << std::endl; }
```

④

```
class A {
public:
void fun4(int a) { std::cout << “void fun4(int a)” << std::endl; }
void fun4(int a) const { std::cout << “void fun4(int a) const” << std::endl; }
};
```

⑤

```
int fun5(int a) { std::cout << “int fun5(int a)” << std::endl; }
void fun5(int a) { std::cout << “void fun5(int a)” << std::endl; }
```

⑥

```
void fun6(int a) { std::cout << “void fun6(int a)” << std::endl; }
void fun6(int a, int b = 1) { std::cout << “void fun6(int a, int b = 1)” << std::endl; }
```

8. 以下代码_____（可以/不可以）正常编译运行，程序输出_____，程序整体返回值为____(无法正常编译运行时后两空填“x”)

```
#include <iostream>
class Base {};
class Derived : public Base {};
double divide(int a, int b) {
    if (b == 0) { throw Derived(); }
    return a * 1.0 / b;
}
int main() {
```

```
    double ans = 0.0;
    try{
        ans = divide(2, 0);
    } catch(const Base& b){
        std::cout << "Base" << std::endl;
        return 1;
    } catch(const Derived& d){
        std::cout << "Derived" << std::endl;
        return 2;
    }
    std::cout << ans << std::endl;
    return 0;
}
```

9. 请阅读以下程序后，回答问题。

```
class Circle {
private:
    double r;
…
};
class Loop {
private:
    double r;
public:
    void display_area(const Circle& c){std::cout<<r*r-c.r*c.r<<std::endl;}
};
```

请问上述 display_area 函数有什么问题？如果使用 C++中的友元来纠正这个问题，应该如何修改上述代码？

10. 请阅读以下程序后，回答问题。

```
class Person {
public:
    virtual void display(){
        std::cout<<“I am a person”<<std::endl;
    }
…
};
```

```
class Student: virtual public Person {
public:
     void display(){
          std::cout<< “I am a student” <<std::endl;
     }
…
};
class Teacher: virtual public Person {
public:
     void display(){
          std::cout<< “I am a Teacher” <<std::endl;
     }
…
};
class StudentTeacher: public Student, public Teacher {
public:
     void display(){
          std::cout<< “I am both a Student and a Teacher” <<std::endl;
     }
…
};
int main() {
    Person* person=new StudentTeacher();
    person->display();
    person->Student::display();
    person->Teacher::display();
    person->Person::display();
    delete person;
    return 0;
}
```

请问 main 函数中的哪些 display 函数可以正常调用，它们的输出分别是什么？

请将第四题第 1 小题的答案，写在 3 号答题纸上。拍照后的文件，命名为 3-1.jpg，如需要加页，加页拍照后的文件命名为 3-1-01.jpg

3 号答题纸　　姓名：　　学号：

第四题编程题

1.

四、程序设计题（30 分，每题 15 分）：

1. Planet 是一个抽象类，Earth 继承了 Planet 并重载了 show 函数，为了使 main 函数正常运行，需要对以下代码进行完善：

```
class Planet{
public:
    virtual void show()=0;
}
class Earth: public Planet{
private:
    char* name;
public:
    void show(){std::cout<<“Welcome to ”<< name << std::endl;}
    Earth(){name=nullptr;}
    Earth(const char* name);
    Earth(const Earth& earth);
    Earth& operator=(const Earth& earth);
    ~Earth();
};
int main(){
    Planet* planet=new Earth(“earth”);
    planet->show();
```

```
    delete planet;
    return 0;
}
```

（1）请补充 Planet 类中析构函数的声明；

（2） 请定义 Earth 类的构造函数：Earth(const char* name)和 Earth(const Earth& earth)；

（3）请定义 Earth 类的析构函数~Earth()；

（5）请定义 Earth 类的赋值“=”运算符重载 Earth& operator=(const Earth& earth)；

请将第四题第 2 小题的答案，写在 4 号答题纸上。拍照后的文件，命名为 4.jpg，如需要加页，加页拍照后的文件命名为 4-1.jpg

4 号答题纸　　姓名：　　学号：

第四题编程题

2.

2.某同学结合课程中多态的相关知识实现了四则运算的简单工厂模式代码，请编程补全其相关实现，满足 main 函数正确输出四则运算结果的效果：

```
int main() {
    int left = 3, right = 5;

    Operation* add = createOperation('+');
    Operation* minus = createOperation('-');
    Operation* times = createOperation('*');
    Operation* divide = createOperation('/');

    double addRes = add->operate(left, right);
```

```
    double minusRes = minus->operate(left, right);
    double timesRes = times->operate(left, right);
    double divideRes = divide->operate(left, right);

    std::cout << "Add result is: " << addRes << std::endl;
    std::cout << "Minus result is: " << minusRes << std::endl;
    std::cout << "Times result is: " << timesRes << std::endl;
    std::cout << "Divide result is: " << divideRes << std::endl;

    delete add;
    delete minus;
    delete times;
    delete divide;

    return 0;
}
```

（1）设计并实现基类 Operation；

（2）设计并实现具体四则运算类 AddOperation、MinusOperation、TimesOperation 和 DivideOperation；

（3）设计并实现根据运算符号返回响应运算类的函数 createOperation；

（4）使用异常保证代码在参数输入有误时抛出合理的异常提示；