

第1题

/*1:对于如下类型声明, 分别给出类A、B、C可访问的成员及其访问权限*/

```
class A {
    int a1, a2;
protected:
    int a3, a4;
public:
    int a5, a6;
};
class B : A {
    int b1, b2;
protected:
    using A::a3;
    int b3, b4;
public:
    using A::a6;
    int b5, b6;
};
struct C : public B {
    int c1, c2;
    using B::a3;
protected:
    int c3, c4;
public:
    int c5, c6;
    int b5, b6, a6;
};
```

```
//A:
a1,a2 : private
a3,a4 : protected
a5,a6 : public
//B
a4,a5,b1,b2 : private
a3,b3,b4 : protected
a6,b5,b6 : public
//C
a3,a6,b5,b6,c1,c2,c5,c6 : public
b3,b4,c3,c4 : protected
```

第2题

/*2: 根据以下代码, 回答问题或完成类的定义*/

```
class A {
    int i;
```

```
    public:
        A(int i);
};
/*2.1 请在class A的类体外实现构造函数A(int i), 要求在成员初始化列表初始化i成员*/

    class B :public A {
        int j;
    public:
        B(int i, int j);
};
/*2.2 请在class B的类体外实现构造函数B(int i,int j)要求在成员初始化列表初始化j成员*/
```

```
//2.1
A::A(int i) : i(i) {}
//2.2
B::B(int i, int j) : A(i), j(j) {}
```

第3题

```
/*3: 根据以下代码, 回答问题或完成类的定义*/
class A {
protected:
    int i{ 0 };
public:
    A(int i = 0) {}
};

class B :public A {
    int i;
public:
/*3.1 B类构造函数没有显式构造基类A对象, 编译是否可以通过? 请说明原因*/
    B(int i) { this->i = i; }
    int sum(int i) {
/*3.2 请完成sum函数的具体实现, 要求对局部变量i、A类继承的i、B类的i求和并返回*/
    }
};
```

```
//3.1
//可以编译通过, 编译器会自动调用 A 类的默认构造函数。
//由于 A 类定义了一个带有默认参数的构造函数 A(int i = 0), 这个默认构造函数会被调用
//3.2
int B::sum(int i) {
    return i + A::i + this->i;
}
```

第4题

/*4: 根据以下代码, 回答问题*/

```
class A {  
    int i;  
public:  
    A(int x) :i(x) { }  
};
```

```
class B {  
    int j;  
public:  
    B() :j(0) {}  
    B(int x) :j(x) { }  
};
```

```
class C :public A {  
    const int k;  
    A a;  
    B b;  
    A& ra;  
public:
```

/*下面类C的构造函数中正确的有哪些? 错误的有哪些? 错误的请说明原因*/

```
    C(int v) :A(v), k(v), a(v), ra(a) {} //1  
    C(int v) :A(v), a(v), ra(a) { k = v; } //2  
    C(int v) :A(v), k(v), a(v), b(v), ra(a) {} //3  
    C(int v) :k(v), a(v), b(v), ra(a) {} //4  
};
```

1--> 正确

2--> 错误, k是 const 类型的成员, 必须在初始化列表中进行初始化

3--> 正确

4--> 错误, 基类 A 没有在初始化列表中进行初始化, 因此会导致编译错误

第5题

/*5:下面程序, 写出指定语句的输出结果, 并解释原因。*/

```
class A {  
public:  
    virtual void f() { std::cout << "A::f()" << std::endl; }  
    virtual void f(double x) { std::cout << "A::f(double)" << std::endl; }  
    static void g() { std::cout << "A::g()" << std::endl; }  
    A() = default;  
    virtual ~A() = default;  
};
```

```

class B :public A {
public:
    virtual void f() { std::cout << "B::f()" << std::endl; }
    virtual void f(double x) { std::cout << "B::f(double)" << std::endl; }
    static void g() { std::cout << "B::g()" << std::endl; }
    static void g(int) { std::cout << "B::g(int)" << std::endl; }
    B() = default;
    virtual ~B() = default;
};

```

/*请说明当执行下面的test函数后，每一条语句的情况，如果可以运行请给出运行结果；如果编译出错请说明原因*/

```

void test1(A& o) {
    o.f();           //语句1
    o.f(1.0);        //语句2
    o.g();           //语句3
    o.g(1);          //语句4
    ((B)o).f();       //语句5
    ((B)o).g();       //语句6
}
void test2(B& o) {
    o.f(1.0f);        //语句7
    o.f(1.0);         //语句8
    o.g(1);           //语句9
    o.g();            //语句10
    ((A)o).g();       //语句11
}
void test() {
    B b;
    test1(b);
    test2(b);
}

```

```

1--> B::f()
2--> B::f(double)
3--> A::g()
4--> 编译出错，静态成员函数不支持多态，A 类中没有 g(int) 函数
5--> B::f()
6--> B::g()

7--> B::f(double)
8--> B::f(double)
9--> B::g(int)
10--> B::g()
11--> A::g()

```

第6题

```
/*6: 下面是类A、B、C的定义*/
class A {
public:
    virtual void fa() {}
    virtual void fb() = 0;
    virtual void fc() = 0;
    A() = default;
    virtual ~A() = default;
};

class B :public A {
public:
    virtual void fb() {}
};

class C :public B {
public:
/*6.1: 如果C想成为一个具体类, 则C必须要实现的方法是什么*/
};

/*6.2: 请指出下面代码有错误的地方, 并说明原因*/
void f(A& o) {}
A& f(A* p) { return *p; }
void f(A o){}
```

6.1--> C 必须实现的方法是 fc()

6.2--> void f(A o) 是错误的, 因为 A 是一个抽象类, 不能实例化

第7题

```
/*7: 请给出下面程序的输出结果*/
class A {
public:
    virtual void draw() { std::cout << "Draw A" << std::endl; }
    virtual void display() { draw(); std::cout << "Display A" << std::endl; }
    A() = default;
    virtual ~A() = default;
};

class B :public A {
public:
    virtual void draw() { std::cout << "Draw B" << std::endl; }
    virtual void display() { A::display(); std::cout << "Display B" <<
std::endl; }
    B() = default;
    virtual ~B() = default;
};
```

```
void test() {  
    std::unique_ptr<A> ptr = std::make_unique<B>(B());  
    ptr->display();  
}
```

```
Draw B  
Display A  
Display B
```

更正

// 第五题		
((B)o).f();	//语句5	编译出错：父类不能强制转换成子类
((B)o).g();	//语句6	编译出错：父类不能强制转换成子类