概念题

1. 简述 C++中继承的概念，并说明继承有哪些优点。

对一个面向对象的程序，在定义一个新的类时，先把已有程序中的一个或多个类的功能全部包含进来，然后在新的类中再给出新功能的定义或对已有类的某些功能重新定义。

支持软件复用，对处理的对象按层次进行分类，对概念进行组合，支持软件增量开发。

2. C++提供几种继承方式？请列举派生类中继承自基类成员的各种访问控制。

三种。public，private，protected。

基类成员 派生类 继承方式	public	private	protected
public	public	不可直接访问	protected
private	private	不可直接访问	private
protected	protected	不可直接访问	protected

3. 简述 C++中转移构造函数与转移赋值函数的作用。

解决用一个临时对象或即将消亡的对象去初始化另一个对象问题，实现资源的转移。

解决用于赋值的对象是一个临时或即将消亡的对象问题，实现资源的转移，提高效率。

编程题

1. 现有一个普通汽车类 Car，升级改造成能够完成自动驾驶的功能的 AutopilotCar 类。

```
#include <iostream>
using namespace std;

class Car
{
protected:
    float fRadian;
    float fSpeed;
    float fDeltaTime;
public:
    void drive(float fRadian, float fSpeed, float fDeltaTime) {
        cout << "Driving..." << endl;
        cout << "Radian: " << fRadian << endl;
        cout << "Speed: " << fSpeed << endl;
        cout << "DeltaTime: " << fDeltaTime << endl;
    }
};
```

```
class AutopilotCar : public Car {
public:
    void autoDrive() {
        cout << "autoDriving..." << endl;
    }
};

class UpgradedAutopilotCar : protected AutopilotCar {
public:
    void autoDrive() {
        AutopilotCar::autoDrive();
    }
    void optimizedDrive() {
        fRadian += rand() / double(RAND_MAX);
        fSpeed += rand() / double(RAND_MAX);
        fDeltaTime -= rand() / double(RAND_MAX);
        drive(fRadian, fSpeed, fDeltaTime);
    }
};

class PerfectCar : private AutopilotCar {
public:
    void autoDrive() {
        AutopilotCar::autoDrive();
    }
};


int main()
{
    Car car;
    car.drive(30.0, 60.5, 2.0);
    AutopilotCar auto_car;
    auto_car.drive(30.0, 60.5, 2.0);
    auto_car.autoDrive();
    UpgradedAutopilotCar up_car;
    up_car.optimizedDrive();
    up_car.autoDrive();
    PerfectCar per_car;
    per_car.autoDrive();
    return 0;
}
```

2. 阅读以下程序，写出其运行结果，并与程序运行的实际结果进行比较。

Base()
A()
Derive()

~Derive()
~A()
~Base()

Base()
A()
Derive()
Base()
A()
Derive()
Base()
A()
Derive(const Derive&)
~Derive()
~A()
~Base()
~Derive()
~A()
~Base()
Derive& operator = (const Derive& testDerive)

~Derive()
~A()
~Base()
~Derive()
~A()
~Base()