//知识点：虚函数和多态

1.阅读并执行代码，然后回答问题

```
class A
{
public:
	A()            { cout<<"Lines="<<++lineno<<" Call A::A()"<<endl;}
	virtual ~A() { cout<<"Lines="<<++lineno<<" Call A::~A()"<<endl;} //----------------(1)
	virtual    int Func1() const=0;          //-----------------------------------------------(2)
	virtual    void Func2(int=500)=0;      //-----------------------------------------------(3)
protected:                          //-------------------------------------------------------------(4)
	static int lineno;
};
class B:public A
{
public:
	B()            {cout<<"Lines="<<++lineno<<" Call B::B()"<<endl;}
	virtual ~B() {cout<<"Lines="<<++lineno<<" Call B::~B()"<<endl;} //----------------(5)
	virtual int    Func1(int n) const {return num+n;}
	virtual int    Func1() const                  //-----------------------------------------(6)
	               {cout<<"Lines="<<++lineno<<" Call B::Func1() const"<<endl;    return num; }
	virtual void Func2(int n=1000)          //-----------------------------------------(7)
	               {cout<<"Lines="<<++lineno<<" Call B::Func2(int) const n="<<n<<endl; num=n;}
	void Func3(int n) {num=n;} //-----------------------------------------(8)
protected:
	int num;
};

int A::lineno=0;                                  //-----------------------------------------(9)
int main(int argc,   char* argv[])
{
	A * p=new B;                                  //-----------------------------------------(10)
	p->Func2(50);                                 //-----------------------------------------(11)
	p->Func2();                                   //-----------------------------------------(12)
	p->Func1();                                   //-----------------------------------------(13)
	//p->Func3(100);                              //-----------------------------------------(14)
	//p->Func1(100);                              //-----------------------------------------(15)
	delete p;
	return 0;
}
```

1)去掉(1)中的 virtual，比较执行结果。

2)去掉(6)中的 const，可以吗？

3)(3)和(7)中各自定义了参数的缺省值，(12)执行时，匹配的是哪个缺省值，为什么？由结果能得到什么结论？

4)(4)中的 protected 改为 private 可以吗？
5)去掉(5)中的 virtual，对结果有影响吗？
6)(9)的作用是什么？
7)(10)中改为 A* p=new A;可以吗？理解抽象类和具体类的概念。
8)去掉(3)中的 virtual，结果会有什么改变？
9)(14)为什么不对？将(8)改为 virtual 后，(14)可以了吗？
10)(15)为什么不对？理解编译时，使用静态类型；运行时，使用动态类型的含义。

2.一个游戏中有多种怪物(Monster)，怪物之间可能要发生战斗(fight)，每场战斗都是一个怪物与另一怪物之间的一对一战斗。每个怪物都有自己的生命值(hitpoint)、攻击力值(damage)和防御力值(defense)，每种怪物都有各自特有的攻击(attack)方式，产生相应的攻击效果；战斗时，两个怪物依次攻击对方，即怪物 a 首先攻击怪物 b， 然后轮到怪物 b 攻击怪物 a，之后，怪物 a 再次攻击怪物 b，…， 直到一方生命值为 0。
请根据你对上述描述的理解，定义并实现怪物类 Monster，成员的设计可以任意，但要求该类至少有一个不带 virtual 修饰的成员函数 fight，用来描述与另外一个怪物进行战斗的过程，该函数的实现可为 Monster 类的任意派生类所复用（派生类不需重新定义及实现）。不必考虑怪物的生命值减少至 0 后如何处理。

3.作为怪物的特例，猫和狗的攻击效果如下表所示。在 Monster 的基础上，以继承手段定义并实现这两个类。
猫进攻导致对方的生命值减少量：
(猫的攻击力值 * 2 - 对方的防御力值) 若上式小于 1，则取 1
狗进攻导致对方的生命值减少量：
(狗的攻击力值 - 对方的防御力值 + 5) * 2 若上式小于 2，则取 2

4.给出适当的类设计和相应的代码。
有一个只能放进不能取出的盒子，最多可放 8 个水果，不一定同一天放入。水果只是苹果和桔子两种，它们放入盒子前的原始重量分别为 50 和 30，放入盒子后，由于丢失水分，它们的重量减轻，苹果和桔子每天分别减轻 4 和 3，直到达到各自原始重量的 3/5 后，不再减轻重量。盒子的功能有：输出盒子中苹果的数量；输出盒子中桔子的数量；输出一天来盒子中水果减轻的总重量；输出当前水果的总重量。(使用虚函数和向下类型转换)

5.小王编写一个程序时，定义了类 B 和全局函数 f：

```
class B {
public:
    B(int n):data(n) { }
    int Data( ) const { return data; }
    void g1( );
    void g2( );
    void g3( );
private:
    const int data;
};
void f( B& b ) {
```

```
    int condition = b.Data( );
    if(condition == 1) { b.g1( ); }
    else if(condition == 5) { b.g2( ); }
    else if(condition == 9) { b.g3( ); }
}
```

当把此程序交给用户试用时，针对函数 f，用户提出了一项新的要求：当 condition 为 100 时，依次执行 b 的成员函数 g1( )和 g2( )。经过进一步了解，小王获悉：以后可能还要增加处理 condition 的值是其它数值时的情况，但这些需要分别处理的不同条件值的个数肯定不多。小王希望他写出的代码既能满足上述要求，又不用每次都改写 f 的代码。请你帮小王重新设计，使得新设计能够满足小王的愿望。简要说明你的设计思想，给出实现代码。

6.阅读代码，并按要求练习。

```
#include <iostream>
using namespace std;
class CallBackObject;
class Server {
public:
      Server(int size) :len(size){
              -----(1)-----;
           for(int i=0;i< len;++i)
                data[i]= i+1;
      }
      ~Server( ) {     ----(2)-----; }
        int Total(CallBackObject& obj);
private:
           int    len;
           int * data;
};
class CallBackObject {
public:
      virtual ~CallBackObject( ) {}
      ----(4)-----;
};
class ClientA:public CallBackObject {
public:
      virtual ~ClientA( ) {}
        virtual int CallBackFunc(int val){
           return val;
        }
    void RequestA(Server& srv) {
         cout<<    -----(5)-----    <<endl;
    }
};
int Server::Total(CallBackObject& obj)
```

```
{
int sum = 0;
        for(int i=0; i<len; ++i) {
              -----(3)-----;
        }
        return sum;
}
class ClientB:public CallBackObject {
public:
      virtual ~ClientB( ) {}
      virtual int CallBackFunc(int val){
          -----(6)-----;
      }
      void RequestB(Server& srv) {
          -----(7)-----;
      }
};
//主函数 1
int main( ) {
   Server    srv2(2), srv5(5);
   ClientA    a;
   a.RequestA(srv2);    //输出 3
   a.RequestA(srv5);    //输出 15
   return 0;
}
//主函数 2
int main( ) {
     Server    srv2(2), srv3(3);
     ClientB    b;
     b.RequestB(srv2);
     b.RequestB(srv3);
     return 0;
}
```

1)请分别给出空格 1-5 中正确的代码，使得主函数 1 的输出为 3 和 15。

2)请分别给出空格 6-7 中正确的代码，使得主函数 2 的输出为：

平方和=5

平方和=14

3)请重新实现 ClientB 中的相关函数，使得主函数 2 的输出为：

1　2　的立方和=9

1　2　3　的立方和=36