**BL1000. Virtual class**

**Description**

设计一个基类base，包含姓名和年龄私有数据成员以及相关的成员函数，由它派生出领导类leader，包含职务和部门私有数据

成员以及相关的成员函数，再由base派生出工程师类engineer，包含职称和专业私有成员以及相关的成员函数。然后由leader

和engineer类派生出主任工程类chairman，采用一些数据测试。

基类：

class base

{

char \*name;

int age;

public:

base(){}

void setname(char arr[]);

void setage(int age);

char\* getname();

int getage();

};

提交时，需要提交类base、leader、engineer、chairman的定义与实现；

主函数如下：

int main()

{

//freopen("in.txt", "r", stdin);

//freopen("out.txt", "w", stdout);

int n;

cin>>n;

for (int i=0;i<n;i++)

{

char name[20],job[20],dep[20],major[20],prof[20];

            int age;

cin>>name>>age>>job>>dep>>major>>prof;

chairman man;

man.setname(name);

man.setage(age);

man.setjob(job);

man.setdep(dep);

man.setmajor(major);

man.setprof(prof);

cout<<"name: "<<man.getname()<<" age: "<<man.getage()<<" dep: "<<man.getdep()

<<" job: "<<man.getjob()<<endl;

cout<<"prof: "<<man.getprof()<<" major: "<<man.getmajor()<<endl;

}

return 0;

}

**Hint**

注意：由于chairmen类是从leader类和engineer类派生而来，而leader类和engineer类都是从base类派生的，此时应该要注意什么问题？

**BL1001. About Polymorphic**

**Description**

下面是不完整的继承类定义，需要补充或增加完整的构造函数、析构函数、void fun()

class A

{

   void g()

   {

      fun();

   }

};

此外该类需要定义函数 fun() ，实现输出：Call class A's fun

class B

{

};

继承来自于基类A，同时补充完整类B

class C

{

};

继承来自类B，同时还有函数fun，使其输出：Call class C's fun

int main()

{

A \*a=new C;

a->g();

delete a;

return 0;

}

输出结果：

A constructor

Call class A's fun

B constructor

C constructor

Call class C's fun

C destructor

B destructor

A destructor

**Input**

 none

**Output**

 none

**Hint**

 只需要提交类A、B、C的定义与实现

**BL1002. Destructor**

**Description**

 下面是不完整的继承类定义:

class A

{

public:

virtual void Prin()

{

cout<<"Prin come form class A"<<endl;

}

};

class B

{

   char \*buf;

public:

     void Prin()

{

cout<<"Prin come from class B"<<endl;

}

};

试完成其定义(你可以根据需要增加必要的构造函数、析构函数 ),使得主函数main运行后能得到其后结果：

void fun(A \*a)

{

delete a;

}

int main()

{

A \*a = new B(10);

a->Prin();

fun(a);

B \*b =  new B(20);

fun(b);

return 0;

}

运行结果：

Prin come from class B

B::~B()called

A::~A()called

B::~B()called

A::~A()called

**Input**

 none

**Output**

 none

**Hint**

virtual destructor!

only need to submit class A, B and main()