**BM1000. Chain Call**

**Description**

 As shown in the following code segment, **op** is an object of class **Number**. Please implement **Number** in which you should:

<!--[if !supportLists]-->1.    <!--[endif]-->write the member functions of **add** and **sub** with one int parameter respectively.

<!--[if !supportLists]-->2.    <!--[endif]-->write a **constructor** with one int parameter.

<!--[if !supportLists]-->3.    <!--[endif]-->write a **print** function to print the result.

Number op(1);

op.add(2).sub(3).add(4)

op.print();   // output：4

**Sample Input**

http://soj.sysu.edu.cn/images/clipboard.jpgCopy sample input to clipboard

Sample Test

Number n(1);

n.add(2).sub(3).add(4);

n.print();

**Sample Output**

4

**BM1001. Area and Perim**

**Description**

声明一个Shape抽象类，在此基础上派生出Rectangle类和Circle类，两者都有GetArea（）和GetPerim（）函数，用来计算面积和

周长，麻烦封装以上各类，编写程序。需要提交类shape、Rectangle、Cricle类

主函数：

int main ()

{

double n,a,b,c;

cin>>n;

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

{

Shape \*s;

cin>>a>>b>>c;

s = new Rectangle(a,b);

cout<<s->GetArea()<<" "<<s->GetPerim()<<endl;

delete s;

s = new Circle(c);

cout<<s->GetArea()<<" "<<s->GetPerim()<<endl;

                delete s;

}

return 0;

}

**Input**

第一行输入n，表示n个测试例子；接下来有n行，每一行输入数，a、b、c，表示矩形的宽、长，以及圆形的半径

**Output**

 对应每一行输入：

第一行输出矩形的面积、周长

第二行输出圆形的面积、周长

**Sample Input**

http://soj.sysu.edu.cn/images/clipboard.jpgCopy sample input to clipboard

1

2 2 3

**Sample Output**

4 8

28.26 18.84

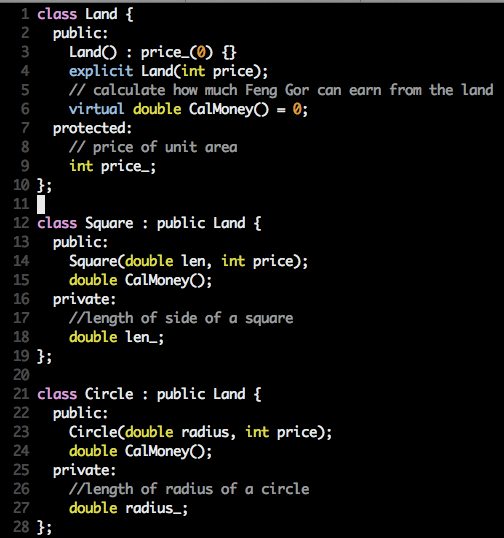
**Hint**

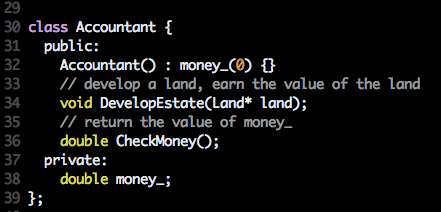
 const double PI = 3.14； (Already defined)

**BM1002. Estate**

**Description**

 Feng Gor是著名的房地产商，她每天的主要工作就是——数钱。这并不是一件容易的事，Feng Gor主要投资两类地，一类是圆的，一类是方的。现在你作为她的会计，需要帮她数钱。Circle类和Square类继承于Land类，分别代表圆的地和方的地，根据单位面积价格(price)和地的面积(根据边长或半径算出)。Accountant类用于计算Feng Gor开发各种房地产带来的收入。DevelopEstate函数接收一个Land的指针，计算开发这块房地产带来的收入（这里用到了动态绑定哦）。CheckMoney函数返回当前的收入。





**Input**

**Sample Input**

http://soj.sysu.edu.cn/images/clipboard.jpgCopy sample input to clipboard

//这里给出一个测试的main函数

#include <iostream>

#include <iomanip>

using namespace std;

int main(int argc, const char \*argv[]) {

Accountant py;

Circle \*a = new Circle(100, 10000);

Square \*b = new Square(100, 50000);

py.DevelopEstate(a);

cout << setprecision(10) << py.CheckMoney() << endl;

py.DevelopEstate(b);

cout << setprecision(10) << py.CheckMoney() << endl;

return 0;

}

**Sample Output**

314159265.4

814159265.4

**Hint**

为了防止大家背不熟圆周率导致Wrong Answer，特意提醒一下，pi可以用acos(-1)表示。acos在cmath库里。

**BM1003. Inservice**

**Description**

 The declaration of a class named Person is shown as follows.

class Person{

public:

    Person(){ cout << "Construct Person" << endl;}

    ~Person(){ cout << "Destruct Person" << endl;}

public:

    string name;      //person’s name

int sex;           //1--male, 0—female

};

Class Student, Worker and InService are direct derived and indirect derived classes of Person. Student has attributes of name, sex, and sno. Worker has attributes of name, sex and wno. sno and wno are the student number and the worker number respectively.

Please use inheritance or virtual inheritance to implement Student, Worker and InService so that they satisfy the following main function.

int main()

{

    InService is;

    is.name = "name";

    is.sex = 0;

    is.sno = "10390000";

    is.wno = "10990000";

    cout << ((Person\*)(&is))->name << endl;

    cout << ((Person\*)(&is))->sex << endl;

    cout << ((Student\*)(&is))->sno << endl;

    cout << ((Worker\*)(&is))->wno << endl;

    return 0;

}

Outputs:

Construct Person

Construct Student

Construct Worker

Construct InService

name

0

10390000

10990000

Destruct InService

Destruct Worker

Destruct Student

Destruct Person

**BM1004. Polymorphism**

**Description**

Feng Gor除了是个房地产商外，像大多数女生一样，她也是喜欢逛淘宝的。我们都知道，在淘宝上有很多很多的商品，当然，Feng Gor挑的一般都是最贵的，但是偶尔也是要看看销量参考一下的啊。因此，她希望淘宝能够按照她想要的方式来排序：当她想要按价格排序，商品就按价格排序；当她想要按销量排序，商品就按销量排序(都是按照从大到小排序)。淘宝这种大公司肯定已经提供这个功能的啊。不过，作为一个程序猿，她希望大家也来用C++模拟实现以下。

代表商品的结构体如下：

struct TaoBaoItem {

  int price;

  int volume\_of\_sales;

};

为了能实现不同的排序方式，她特意抽象出了一个接口出来，

class SortInterface {

 public:

  virtual ~SortInterface() {};

  virtual void DoSort(TaoBaoItem item[], int size) = 0;

};

接下来，她要你们实现的类如下：

class SortByPrice : public SortInterface {

 public:

  virtual void DoSort(TaoBaoItem item[], int size);

};

class SortBySales : public SortInterface {

 public:

  virtual void DoSort(TaoBaoItem item[], int size);

};

class TaoBao {

 public:

  TaoBao(SortInterface\* strategy);

  void SetSortStrategy(SortInterface\* strategy);

  // Use current strategy\_ to do the sort.

  void sort(TaoBaoItem item[], int size);

 private:

  SortInterface \*strategy\_;

};

注意看测试框架的代码！

**Sample Input**

http://soj.sysu.edu.cn/images/clipboard.jpgCopy sample input to clipboard

#include <iostream>

#include <string>

using namespace std;

struct TaoBaoItem {

int price;

int volume\_of\_sales;

};

class SortInterface {

public:

virtual ~SortInterface() {};

virtual void DoSort(TaoBaoItem item[], int size) = 0;

};

#include "source.cpp"

void printItem(TaoBaoItem arr[], int size) {

for (int i = 0; i < size; ++i) {

cout << arr[i].price << " " << arr[i].volume\_of\_sales << endl;

}

}

int main(int argc, char \*argv[]) {

TaoBaoItem item[4] = {

{1, 2},

{2, 3},

{5, 1},

{3, 10}

};

SortByPrice price;

SortBySales sales;

TaoBao taobao(&price);

taobao.sort(item, 4);

printItem(item, 4);

taobao.SetSortStrategy(&sales);

taobao.sort(item, 4);

printItem(item, 4);

return 0;

}

**Sample Output**

5 1

3 10

2 3

1 2

3 10

2 3

1 2

5 1

**Hint**

这其实是设计模式中的一种！（其实这个好像只是个简单的演示代码）叫做策略模式，有兴趣的同学可以在自行Google找到维基百科看，不过，这个题目最主要的目的是想让大家体验一下多态带来的好处。另外，关于排序的实现，大家可以尝试使用stl库中提供的sort函数，自行Google用法啊！