6.5元

上 海 交 通 大 学 试 卷（A）

（ 2010 至 2011 学年 第 二 学期 期末考卷）

班级号_____ 学号_____ 姓名 _____

课程名称 **C++程序设计** 成绩 _____

1．如果>被重载成友元函数，则表达式obj1 > obj2被C++编译器解释为：__A__。

A． operator>(obj1,obj2) B． >(obj1,obj2)

C． obj2.operator>(obj1) D． obj1.operator>(obj2)

2．对于如下定义的两个类模板，__C__中的模板实例化是无效的。

template <class T, int size> class Array { /*.....*/ }

template <int hi, int wid> class Screen { /*.....*/ }

A． const int hi = 40; const int wi = 80; Screen <hi, wi + 32> Sobj;

B． const int arr_size = 1024; Array<string, arr_size> a1;

C． double db = 3.1415; Array <double, db> a3;

D． Array <double, 12> a3;

3．对于拷贝构造函数和赋值操作的关系，正确的描述是__B__。

A. 进行赋值操作时，会调用类的构造函数。

B. 当调用拷贝构造函数时，类的对象正在被建立并被初始化。

C. 拷贝构造函数和赋值操作是完全一样的操作

D. 拷贝构造函数和赋值操作不能在同一个类中被同时定义

4．如果A是已经定义好的一个类，有如下定义语句：A *p[5];则当类对象数组指针p离开它的作用域时，系统自动调用类A的析构函数__C__次。

A. 5 B. 1 C. 0 D. 10

5．继承的主要目的是__D__。

A。增加数据成员 B。增加成员函数 C，实现函数的重载 D。实现代码重用

6．当使用ifstream流类定义一个流对象并打开一个磁盘文件file时，与语句：ifstream infile; infile.open("file")；等价的文件的打开方式为__C__。

A. ofstream infile; infile.open("file"); B. ifstream infile; infile.open("file", ifstream::out);

C． ifstream infile("file"); D． fstream infile; infile.open("file");

7．下列对派生类的描述中错误的是__A__

A. 基类中成员访问权限继承到派生类中都保持不变。

B. 派生类至少有一个基类。

C. 一个派生类可以作为另一个派生类的基类。

D. 派生类的成员除了自己定义的成员外，还包含了它的基类成员。

我承诺，我将严格遵守考试纪律。

承诺人：_____

题号			
得分			
批阅人(流水阅卷教师签名处)			

8．在 public 继承的情况下，派生类对象对基类中的 public 成员、protected 成员和 private 成员的访问特性是__A__。

A. 只有 public 成员可以访问

B. 只有 private 成员不可以访问

C. public 成员和 protected 成员可以访问

D. 三种成员都可以访问

9．在创建派生类对象时，构造函数的执行顺序是__D__

A. 对象成员构造函数，基类构造函数，派生类本身的构造函数。

B. 派生类本身的构造函数，基类构造函数，对象成员构造函数。

C. 基类构造函数，派生类本身的构造函数，对象成员构造函数。

D. 基类构造函数，对象成员构造函数，派生类本身的构造函数。

10．下列描述中，不属于抽象类特性的是__D__。

A. 可以声明虚函数

B. 可以重载构造函数

C. 可以定义友元函数

D. 可以定义其对象

二．看程序，写结果（每题 5 分，共 40 分）

1、给出下面程序运行的结果

```
class A
{ public:
    A(int i, int j)   { a = i; b = j;}
    void move(int x, int y) { a += x; b += y;}
    void show() {   cout << " ( " << a << "," << b<< ")" << endl;}
  private:
    int a, b;
};
class B: public A
{ public:
    B(int i, int j, int k, int l) : A(i, j)   { x = k; y = l;}
    void show() { cout << x << "," << y << endl; }
    void f1() {A::show(); }
  private:
    int x, y;
};

void main()
{
  A e (1, 2);
  B d(3, 4, 5, 6);
  e.show();
  d.move(2, 3);
  d.show();
  d.f1();
}
```

2、写出下列程序的运行结果

```
class Student
{   friend Student fun(Student & rs);
  public:
    Student(int i = 0, double j = 0) :num(i), score(j)
        {cout << "构造 （" << i << ", " << j << "） " << endl; }
    void print() {cout << num << "," << score << endl;}
  private:
    int num;
    double score;
};

Student fun(Student & rs)
{   rs.num = 1100;
    rs.score = 80;
    return Student(rs.num +10, rs.score);
}

void main()
{ Student s1(1000, 90),   s2;

 s1.print();   s2.print();
 s2 = fun(s1);
 s1.print();   s2.print();
}
```

3。写出下面程序的执行结果

```
int main()
{    char ch;
     int ct1 = 0, ct2 = 0;

     cin >> ch;
     while (ch != 'q')     {     ct1++;          cin >> ch;      }
     while ((ch = cin.get()) != 'q')    ct2++;

     cout << "ct1 = " << ct1 << "; ct2 = " << ct2 << endl;
     return 0;
}
```

如果输入如下，则该程序将打印什么内容？

I see a q<Enter>

I see a q<Enter>

其中，<Enter>表示按下回车键

4．写出下面程序的执行结果

```
class up
{ public:       up() {   cout << "It is up" << endl; }
};
class down
{ public:       down() { cout << "It is down" << endl;}
};
int f(int i) throw(up, down)
{    switch(i) {   case 1: throw up();
                           case 2: throw down();
                           default: return i;
                      }
}
int main()
{   for (int i = 1; i <= 3; i++) try { cout << f(i) << endl; }
            catch (up) { cout << "up catched" << endl;}
            catch (down) {cout << "down catched" << endl;}
     return 0;
}
```

5.写出下面程序执行结果

```
template <class T>
class Base
{    T num1;
public:
     Base(T k) {    num1 = k; }
     virtual void work() {     cout << "Base: " << num1 << endl;      }
};
template <class T1, class T2>
class Derived: public Base<T1>
{    T2   num2 ;
public:
     Derived(T1 m, T2 n) : Base<T1>(m) { num2 = n ;}
     void work(){Base<T1>::work(); cout<<"Derived:"<<num2<<endl;    }
};
int main()
{    Base<int> d1(1);
     Derived<char, double> d2('2', 7.8);
     Base<char> *bp = &d2, b2 = d2;
     d1.work(); d2.work(); bp->work(); b2.work();
     return 0;
}
```

6. 找出下列代码中的错误，给出出错行号，并说明错误原因。

```
1    class CConst {
2          int m_data;
3          const int len = 10;
4          static const int size = 10;
5     public:
6          CConst(int d = 0, int l = 0 , int s =0 );
7          void SetData(int d) {     m_data = d;    }
8          void Print() const    {     cout << m_data++ <<endl;    }
9    };
10   CConst::CConst(int d, int l, int s)
11   {     m_data = d;
12         len = l;
13         size = s;
14   }
15   void main()
16   {     const CConst C;
17         C.SetData(4);
19   }
```

7. 写出下列程序执行结果

```
class Test {
    friend ostream &operator<<(ostream &os,    const Test &obj)
     {     os << obj.len << " " << obj.size <<endl;    return os;    }
  public:
     Test(int d = 0)    { len = size;    size += d;    }
     void operator++() { ++len; }
     void operator++(int n) { ++size; }
     operator int() const { return len + size; }
  private:
     int len;
     static int size;
};
int Test::size = 0;
void main()
{   Test    tt1(3),    tt2(5) ;
    cout << tt1 << tt2;
    ++tt1; tt2++;
    cout << tt1 << tt2;
    cout << tt1 + tt2 << endl;
}
```

8. 写出下列程序的执行结果

```
class CBase {
    protected:
      int b_data;
    public:
      CBase(int i) { b_data = i; }
      CBase operator+(const CBase &other) const
          { return b_data + other.b_data ;  }
      Virtual void print() { cout << "CBase::b_data=" << b_data << endl; }
     };

class CDerived:public CBase
{   int d_data;
  public:
      CDerived(int i = 0): d_data(i), CBase(i-4) { }
      void print()
        { cout << "(" << b_data  << ",  " << d_data << ")" << endl; }
      CDerived operator+(const CDerived &other) const
          { CDerived tmp;
            tmp.b_data = b_data + other.b_data;
            tmp.d_data = d_data + other.d_data;
            return tmp;
          }
};

int main()
{   CDerived d1(20), d2(18);
    CBase *p1 = &d1, *p2 = &d2;
    p1->print(); p2->print();
    (*p1 + *p2).print();
    (d1 + d2).print();
    return 0;
}
```

三． 程序填空（每空 1 分，共 20 分）

1．下面程序的执行结果为

1 2 3 4 5 6 7 8 9 下标越界

请填空。

```
class Myexception
{
 public:
      void what() { cout << _____________ << endl;   }
};

int test(int a[], int n, int c);

int main()
{
    int data[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

    for (int i = 0; ; ++i) {
        __________   {
            cout << test(data, 9, i) << '\t';
        }
        catch (Myexception _____________)   { ex.what(); break; }
    }

    return 0;
}

int test(int a[], int n, int c)   ____________________
{
    if (c >= n) _________________;

    ____________________;
}
```

2．下面的函数打开一个保存着一批字符串的文本文件，字符串间用空格分离，各字符串的长度均小于 80。文件名作为参数传入函数。函数读出文件中的字符串，并输出字符串到屏幕，每行一个，最后统计输出在文件中一共有多少个字符串。

```
#include "iostream"
_____________________________________________
using namespace std;

void FR(char* fname)
{
_____________ infile(fname);
 char a[80];
 int count =_____________;

 if  ( _____________) {
    cout << "can't open file";
    return;
 }
 while (_____________ ) {
     cout << a << endl;
     _____________;
 }
 _____________    ;
 cout << "字符串总数为 " <<   count   << endl;
}
```

3．程序运行结果如下：

```
Student: 1, 张三 grade: 2
Teacher: 1, 李四, Professor
```

请填空。

```
class Person
{
public:
    Person(int pid, char *n)
    {
        ID=pid;
        _____________________________________;
        strcpy(name, n);
    }
    _____________________ ;
    virtual ~Person() { delete name ; }
protected:
    char *name;
```

```
    int ID;
};

class Student: public Person
{
public:
    Student(int sid, char *sn, int grd) :_____{ grade = grd; }
    void print()
    {
      _____;
    }
private:
    int grade;
};

class Teacher: public Person
{
public:
    Teacher(int tid, char *tn, char *pos) : _____

        {  _____;}
    void print()
    {
      _____
    }
private:
    char position[10];
};

int main()
{
    Person *p;
    p=new Student(1, "张三", 2);
    p->print();
    p=new Teacher(1, "李四", "Professor");
    p->print();
    return 0;
}
```

四. 编程（共 30 分）

1．(15 分)设计并实现一个 IntArray 类。这个类保存了一组按递增次序排列的整型数，所需完成的功能如下：

1). 定义一个对象，定义时用户指定该对象最多允许保存的整型数个数，个数的缺省值为 100。

2）添加一个数据，并保持有序性。

3）删除一个数据，并保持有序性。

4）返回对象中第 k 小的元素（用重载下标运算符实现。例如对象为 a，a[k]是第 k 小的元素）。

5）输出对象中的所有元素（用重载<<实现）。

评分标准：

类定义　5 分

构造函数　2 分

添加函数　2 分

删除函数　2 分

下标运算符重载函数　2 分

输出运算符重载函数　2 分

2．（10 分）建立一个基类 Building ，用来存储一座楼房的层数、房间数以及它的总平方英尺数。建立派生类 Housing，继承 Building，并存储下面的内容：卧室和浴室的数量，另外， 建立派生类 Office,继承 Building，并存储灭火器和电话的数目。每个类都必须有设置和获取所保存信息的功能。

评分标准：

基类　4 分

两个派生类　各 3 分

3．（5 分）设计一个动态的、安全的二维数组类模板 Matrix。可以通过

Matrix<int> table(3, 8);

定义一个 3 行 8 列的二维数组，通过 table（i, j）访问 table 的第 i 行第 j 列的元素。例如：table（i, j）= 5；　或 table（i, j）= table（i, j+1）+ 3；行号和列号从 0 开始。

评分标准：

正确设计数据成员　1 分

构造函数　2 分

析构函数　1 分

函数调用运算符重载函数　1 分

上 海 交 通 大 学 试 卷（A）

（ 2009 至 2010 学年 第 二 学期）

1、类 CStudent 的拷贝构造函数的声明语句为__D__。

A．CStudent &CStudent (const CStudent other)

B．CStudent CStudent(const CStudent other)

C．CStudent (CStudent *other)

D．CStudent (const CStudent &other)

2、类的友元函数能访问该类的__C__。

A．私有成员 B．保护成员 C．所有成员 D．公有成员

3、下面关于静态数据成员的描述中，正确的是__A__。

A．静态数据成员可以直接用类名调用

B．静态数据成员可以在类体内进行初始化

C．静态数据成员不能受 private 控制符的作用

D．类的不同对象有不同的静态数据成员值

4、当使用 fstream 流类定义一个流对象并打开一个磁盘文件时，文件的隐含打开方式为__C__。

A．ios::in B．ios::out C．fstream::in | fstream::out D．没有指定打开方式

5、若 char p[20]="hello world";则输出该字符串正确的语句是__D__。

A．cout<<p[20]; B．cout<<&p; C．cout.<< *p; D．cout << p;

6、在派生类中重新定义虚函数时，除了__B__方面，其他方面都必须与基类中相应的虚函数保持一致。

A．参数个数和类型 B．函数体 C．函数名称 D．返回类型

7、类模板定义如下：

template <class T, int low, int high>

class Array {….};

对该类模板实例化正确的是__A__。

A．Array<float, 0, 20> x; B．Array<int, int, int> x;

C．template<int, 0, 20> x; D．Array<int, 0, int> x;

8、公有成员提供了类对外部的接口，私有成员是类的内部实现，而__D__不许外界访问，但允许派生类的成员访问，这样既有一定的隐藏能力，也提供了开放的接口。

A．公有成员 B.私有成员 C.私有成员函数 D.保护成员

9、假定 AB 为一个类，则执行 AB a(2), b[3], *p; 语句时共调用该类构造函数的次数为__C__。

A. 1 B. 3 C. 4 D. 5

10、如果 A 是已经定义好的一个类，函数 f 的原型为 A f(). r2 是 A 类的一个对象，在函数 f 中执行 return r2 时，系统将自动调用__B__。

A. 缺省的构造函数 B. 拷贝构造函数 C. 赋值运算符重载函数 D. 不调用任何函数

一. 看程序，写结果（每题 5 分，共 35 分）

1、请写出下列程序运行结果

```
class ADD
{   friend ADD operator++(ADD op);
    friend ADD operator++(ADD &op, int n);

public:
    ADD(int i = 0, int j = 0) {a = i; b = j;}
    void Show() const {cout << "a=" << a << ",b=" << b << endl;}
private:
    int a, b;
};
ADD operator++(ADD op)
{   ++op.a;  ++op.b;  return op;}

ADD operator++(ADD &op, int n)
{   ++op.a;  ++op.b;  return op;}

void main()
{   ADD obj(1, 2);
    obj.Show();   (obj++).Show();   obj.Show();
    (++obj).Show();   obj.Show();
}
```

```
a=1, b=2
a=2, b=3
a=2, b=3
a=3, b=4
a=2, b=3
```

2、请写出下列程序运行结果

```
class CConAndDecon {
public:
    CConAndDecon(char value) {
        m_data =   value;
        cout << "Object " << m_data <<" constructor" <<endl;
    }

    CConAndDecon(const CConAndDecon &other) {
        m_data =   other.m_data - 1 ;
        cout << "Object " << m_data <<" copy constructor" <<endl;
    }

    CConAndDecon operator=(const CConAndDecon &right) {
        if(  this !=   &right )
         {  m_data =   right.m_data + 1 ;
            cout << "Object 's new value   is " << m_data <<" " <<endl;
         }
        return *this;
```

```
    }

    ~CConAndDecon()
    {   cout <<"Object " << m_data << " destructor" << endl; }

private:
    char m_data;
};
void Func(CConAndDecon x);
int main()
{   CConAndDecon *p = new CConAndDecon('h');
    static CConAndDecon c1( 'k');
    Func(*p);       delete p;
    return 0;
}
void Func(CConAndDecon x)
{   static CConAndDecon c1 = x;
    CConAndDecon c2 = c1;
}
```

```
Object h constructor
Object k constructor
Object g copy constructor
Object f copy constructor
Object e copy constructor
Object e destructor
Object g destructor
Object h destructor
Object f destructor
Object k destructor
```

3、请写出下列程序运行结果

```
class CMake
{public:
    CMake(int n)
    {   m_data = n ;
        cout << "构造 " << m_data << endl;
    }
    CMake(const CMake &obj)
    {   m_data = obj.m_data + 1;
        cout << "拷贝构造" << m_data << endl;
    }
    ~CMake() { cout << "析构 " << m_data << endl;}
    operator int() const { return m_data; }
private:
    int m_data;
};
CMake   MakeObject( int n )
{   CMake   p (n);
    return   p;
}
int main()
{   cout << MakeObject( 7 ) << endl;
    return 0;}
```

```
构造 7
拷贝构造 8
析构 7
8
析构 8
```

4、请写出下列程序运行结果

```
class BaseFly
{ public:
    virtual void Fly() { cout << "\n----Class BaseFly::Fly()----\n"; }
};
class BirdFly: public BaseFly
{ public:
    void Fly() { cout <<   "----Class BirdFly::Fly()----\n" ;}
};
class DragonFly: public BaseFly
{ public:
    void Fly() { cout << "\n----Class DragonFlyFly::Fly()----\n"; }
};
void main()
{   BaseFly *pBase, oBase;
    BirdFly *pBird = new BirdFly();
    pBase = pBird;
    cout << " \nBirdFly->";
    pBase->Fly();
    DragonFly   *pDragon = new DragonFly();
    pBase = pDragon;
    oBase = *pDragon;
    pBase->Fly();
    pBird->Fly();
    pDragon->Fly();
    oBase.Fly();    }
```

```
BirdFly->----Class BirdFly::Fly()----
----Class DragonFlyFly::Fly()----
----Class BirdFly::Fly()---

----Class DragonFlyFly::Fly()----

----Class BaseFly::Fly()---
```

5、写出下列程序执行结果

```
class   Point
{
    friend bool operator!=(const Point &p1, const Point &p2)
    { return p1.x+p1.y  !=   p2.x+p2.y ;   }

private:
    int   x,   y;

public:
    Point(int a = 1, int b = 1)
    {    x = a; y = b;
         cout << "构造 Point(" << x << "," << y << ")" << endl;
    }
    Point(const Point &p)
    {    x = p.x ; y = p.y ;
```

```
        cout <<   "拷贝构造 Point(" << x << "," << y << ")" << endl;
    }
    ~Point() { cout << "析构 Point(" << x << "," << y << "） " << endl; }
    Point &operator++()
    {     if (x < y)   ++x;
          else   ++y;
          return *this;
    }
    void show() { cout << " Point(" << x << "," << y << ")" << endl; }
    int getx() const {return x ;}
    int gety() const {return y ;}
};

int main( )
{
    const Point origin(10, 5);
    Point point2(6, 7);
    int n = 0;
    while (point2 != origin)   {++point2; ++n;}
    point2.show();
    cout << "n= " << n << endl;
    return 0;
}
```

```
构造 Point（10,5）
构造 Point（6,7）
Point(7,8)
n= 2
析构 Point（7,8）
析构 Point（10,5）
```

6、请写出下列程序运行结果

```
void func( int ) ;

int main()
{    for (int i = 30; i > 0; i /= 3)
         try { func(i);
               cout << "i = " << i << endl;
         }   catch( int) { cout <<   "exception:   int " << endl;   }
             catch (double)  { cout <<   "exception: double " << endl;   }
    return 0;
}

void func(int num )
{    if   ( num % 3 )   throw 3;
    else if   ( num % 5 ) throw 5.5;
}
```

```
i = 30
exception:   int
exception: double
exception:   int
```

7、写出下列程序的输出结果

```
template <class T>
```

```
class Sample
{protected:
    T  n;
public:
    Sample(T i) { n = i; cout << ″construct ″ << n << endl;}
    ~Sample() { cout << ″destruct ″ << n << endl;}
    void disp(){cout << "n=" << setfill ( '#' ) << setw ( 10 ) << n << endl;}
};
template <class T >
class model: public Sample<T>
{   T  m;
public:
    model(T  t1, T  t2): Sample<T>(t1) {m = t2; cout << ″construct ″ << m<< endl; }
    ~model() { cout << ″destruct ″<< m<< endl; }
    void disp(){cout << "n=" << setfill ( '#' ) << setw ( 10 ) << n << ' ' << m << endl;}
    operator T () const { return n + m;}
};
int main()
{   model<int> s (20, 30);
    s.disp();
    cout << (int) s << endl;
    return 0;
}
```

```
construct 20
construct 30
n=########20 30
50
destruct 30
destruct 20
```

三． 程序填空（每空 2 分，共 30 分）

1、下列程序的输出是：

2
5 7
14

请填空。

```
class CConst {
public:
    CConst(int d = 0) : __len(d)___  {   size += d;  }
    void Print() __const____ {  cout << len << " " << size <<endl;  }
    _static_____void show() {   cout << size << endl;  }
private:
    const int len;
    static int size;
};

int CConst::size = 2;_____________________
int main()
{   CConst::  show();
```

```
    const CConst c(5）;
    c.Print();
    CConst c2(7);
    c2.show();
    return  0;
}
```

2、下面是处理二维平面上线段的类，其中的 Point 是上一大题第 5 题中定义的 Point 类，请填空。

```
class line {
    Point start;
    Point end;
public:
    line(int sx, int sy, int ex, int ey):  start(sx,sy), end(ex, ey)  {}
    int length() // 计算线段的长度
   {
     return   sqrt((end.getx()-start.getx()* (end.getx()-start.getx()) +(end.gety()-start.gety()) * (end.gety()-start.gety())  ;
    }
};
```

3、下面的函数打开一个保存着一批字符串的文本文件，字符串间用空格分离，各字符串的长度均小于 20。文件名作为参数传入函数。函数读出文件中的字符串，并输出字符串到屏幕，每行一个，最后统计输出在文件中一共有多少个字符串。

```
void FR(char* fname)
{
    ifstream   fin(fname);
    char a[20];
    int cnt = 0;

    if  (  !fin  ) {
        cout << "can't open file";
        return;
    }
    while (  fin >> a  ) {
        cout << a << endl;
        ++cnt  ;
    }
    fin.close()  ;
    cout << "字符串总数为 " << cnt << endl;
}
```

4、补充函数 equal 使得程序结果为：

```
    5 + 6 = 11;
```

1.111 + 2.222 = 3.333

```
template <class T>
T add ( T x, T y )
{ return x + y ; }

int main()
{   int i = 5, j = 6;
    double y1 = 1.111222, y2 = 2.22222222222;

    cout << i <<" + " << j <<" =" add(i, j) << endl;
    cout << setprecision(4) << y1< " + " << y2 <<" = " << add(y1, y2) << endl;
    return 0;
}
```

四. 编程（共 25 分）

1、设计一个学生类 student，包括姓名和三门课程成绩，利用重载运算符"+"将所有学生的成绩相加放在一个对象中，再对该对象求各门课程的平均分。即，运行下面测试程序时，能得到相应的执行结果。（10 分）

```
void main()
{
    student s1("Li", 78, 82, 86), s2("Zheng", 75, 62, 89);
    student s3("Ma", 89, 87, 95), s4("Xu", 54, 78, 66), s;
    cout << "输出结果" << endl;
    s1.disp(); s2.disp(); s3.disp(); s4.disp();
    s=s1 + s2 + s3 + s4; // 调用重载运算符
    avg(s, 4); // 友元函数求平均分
}
```

本测试程序的执行结果如下：

输出结果：

Li 78 82 86

Zheng 75 62 89

Ma 89 87 95

Xu 54 78 66

平均分 74 77 84

评分标准：

正确定义数据成员（2 分）：姓名和三门课的成绩

正确定义成员函数（3 分）：构造函数，disp 函数，avg 函数

正确定义+重载函数（1 分）：定义为成员函数或友元函数

每个函数的实现：各 1 分

2、编写一个程序，计算扇形面积和球体表面积。

已知：圆周率 = 3.1415926 且并定义为所有对象共享的常量

扇形面积 = 圆周率*半径的平方*角度/360

球体表面积 =4*圆周率*球体半径的平方

要求：你需要从一个抽象类 container 出发，完成对扇形类（sector）和球体类（sphere）的设计。（10 分）

需通过如下的 main 函数：

```
#include <iostream>
using namespace std;
int main()
{   container *bptr;        //定义抽象类指针 bptr
    sphere s_obj(4);        //创建球体对象 s_obj，半径为 4;
    sector c_obj(2 ,270);   //创建圆柱体对象 c_obj，半径为 2，角度为 270 度

    bptr  =  &s_obj;
    cout  <<  "球体表面积："  << bptr->area()  <<  endl;
    bptr  =  &c_obj;
    cout  <<  "扇形面积："  << bptr->area()  <<  endl;
}
```

得到的屏幕输出应为：

球体表面积：201.062

扇形面积：9.423

评分标准：

正确定义基类（2 分）：一个全局共享的常量 PI 和纯虚函数 area

正确定义扇形类（2 分）

正确定义球类（2 分）

正确实现各成员函数：每个成员函数 1 分

3、整型数组 int a[10]中随机地存放有数字 0～9（数字可以重复）。

现请设计一个类，它的功能是在数组 a 中顺序地抽取 5 个数字，使这 5 个数字组成的 5 位数为最大(注意:这 5 个数字的先后顺序必须同其在原数组 a[10]中的先后顺序相同)，返回这个五位数。（5 分）

类的定义如下：

```
class max {
      int data[10];
 public:
      max(int *a);
      int result();
};
```

如果数组 a[10] = { 4, 7, 8, 0, 8, 6, 2, 4, 9, 1 }，定义对象 max m(a)，并执行语句 a.result(), 则返回值为 88691。请补充构造函数和 result 函数。

评分标准：构造函数的实现　1 分

result 函数的实现　4 分

上 海 交 通 大 学 试 卷（A）

（ 2010 至 2011 学年 第 二 学期）

1、类 Sample 的拷贝构造函数的声明语句为__C__。

A．Sample (Sample other)

B．Sample Sample (Sample other)

C．Sample (const Samplet &other)

D．Sample Sample (const Sample &other)

2、Sample 是用户定义的某个类，obj 是 Sample 类的对象，p 是 Sample 类的指针，则执行语句 p = new Sample 时会调用__A__函数，执行 obj = *p 时会调用__C__函数，执行 delete p 是会调用__B__函数。

A．Sample 类的构造函数

B．Sample 类的析构函数

C．Sample 类的赋值运算符重载函数

D．Sample 类的拷贝构造函数

3、对于下面定义的类

```
class Base   {
   protected:   int x;
   public:      Base(int val = 1) { x = val; }
                virtual void disp() {cout << x << endl; }
                void print() {cout << x << endl; }      };
class Derived: public Base { int y;
   public:
                Derived(int val1 = 0, int val2 = 0) : Base(val1) { y = val2; }
                void disp() { cout << "x=" << x << "   y=" << y << endl; }
                void print() { cout << "x=" << x << "   y=" << y << endl; }    };
```

有定义 Derived dd(3, 4);

Base *bp = &dd, bb = dd;

则 dd.disp()执行的是__A__，dd.print()执行的是__B__，bp->disp()执行的是__A__，bb.disp()执行的是__C__。

A、派生类的 disp 函数 B、派生类的 print 函数

C、基类的 disp 函数 D、基类的 print 函数

4、公有成员提供了类对外部的接口，私有成员是类的内部实现，而__C__不许外界访问，但允许派生类的成员访问，这样既有一定的隐藏能力，也提供了开放的接口。

A．私有成员 B. 私有成员函数 C. 保护成员 D. 公有成员

5、如果 A 是已经定义好的一个类，函数 f 的原型为 A f(A &other). r2 是 A 类的一个对象，执行函数调用 f(r2)时会调用__D__，在函数 f 中执行 return r2 时，会调用__A__。

A. 拷贝构造函数 B. 缺省的构造函数 C. 赋值运算符重载函数 D. 不调用任何函数

6、假定要对类 X 定义加号操作符重载成员函数，实现两个 X 类对象的加法，并返回相加后的结果，则该成员函数的声明语句为 __D__

A. X operator+(const X & a , const X & b);　　B.　X　& operator+( );

C. operator+(X　a);　　D.　X operator+(const X & a) const;

7、如Base的定义如第3题所示，则执行了语句Base obj[4] = { 3, 4};时，构造函数被调用了__D__次，obj[0]的x值为__C__，obj[1]的x值为__D__，obj[2]的x值__A__，obj[3]的x值为__A__。

A、1　　B、2　　C、3　　D、4

8、链表结点的结构类型为 struct linkRec {int data; linkRec *next;}，如果指针 rear 指向尾结点，将节点 p 链入表尾，并将 p 作为新的表尾可用语句__C__

A、 rear->next=p->next; rear=p;　　B、rear->next= rear; p->next= p;

C、 rear->next=p; rear=p;　　D、 (*rear).next= rear; (*p).next =p;

9．对友元（friend）不正确的描述是：__D__。

A.　友元关系既不对称也不传递。

B.　友元声明可以出现在 private 部分，也可以出现在 public 部分。

C.　整个类都可以声明为另一个类的友元。

D.　类的友元函数必须在类的作用域以外被定义。

10。关于纯虚函数和抽象类的描述中，错误的是__C__。

A。纯虚函数是一种特殊的虚函数，它没有具体的实现。

B。抽象类是指具有纯虚函数的类。

C。一个基类说明中有纯虚函数，该基类的派生类不再是抽象类。

D。抽象类只能作为基类来使用，其纯虚函数的实现由派生类给出。

二．看程序，写结果（每题 5 分，共 40 分）

1、写出下列程序的执行结果

```
class Sample{
   private:    int x;
   public:     Sample(int val = 0)
                  { x = val; cout   << "构造" << x << endl;}
               Sample(const Sample &obj)
                  { x = obj.x; cout   << "拷贝构造" << x << endl;}
               ~Sample(){cout   << "析构" << x << endl;}
               void operator++() { x++; }
};

void foo(Sample i);

int main()
{
   Sample s1, s2(1);
   foo(s1);
   foo(2);

   return 0;
```

S3 = i+1 时的答案	S3 = i 时的答案
构造 0	构造 0
构造 1	构造 1
拷贝构造 0	拷贝构造 0
拷贝构造 1	拷贝构造 0
析构 0	析构 0
构造 2	构造 2
析构 2	析构 2
析构 1	析构 1
析构 0	析构 0
析构 3	析构 2

```
}
void foo(Sample i)
{   static Sample s3 = I + 1;
    ++s3;
}
```

2．请写出下列程序在执行时会出现什么问题，如何改正这个错误

```
#include <iostream.h>
#include <cstring>

class sample {
private:
   char *string;

public:
   sample(const char *s)    {
      string = new char[strlen(s)+1];
      strcpy(string, s);
   }
   ~sample() { delete string; }
};

sample f(char *arg)
{  sample tmp(arg);
   return tmp;
}
int main()
{  sample local = f("abcd");

   return 0;
}
```

程序执行会异常终止。
原因是缺少拷贝构造函数使得 local 的 string 指向的空间不存在

3、请写出下列程序运行结果

```
class ADD
{
   friend bool operator<=(const ADD &p1, const ADD &p2)
        { return p1.a+p1.b  <=  p2.a+p2.b ;  }

   public:
        ADD(int i = 0, int j = 0)
            { a = i; b = j; }
        void Show() const
            { cout << "a=" << a << ",b=" << b << endl; }
        ADD &operator++()
            { ++a;  return *this;  }
        ADD operator++(int n) ;
   private:
        int a, b;
};
```

```
ADD ADD::operator++( int n)
{
    ADD   tmp = *this;
    ++b;
    return tmp;
}

int main( )
{
    const ADD origin(10, 6);
    ADD value(6, 7);
    int n = 0;
    while (value <= origin)
        if ( n++ % 2) ++value; else value++;
    value.Show();
    cout << "n= " << n << endl;

    return 0;
}
```

a = 8, b = 9
n = 4

4、请写出下列程序运行结果

```
class Base   {
  protected:  int x;
  public:     Base(int val = 0) { x = val; }
              virtual void operator++() { x++; }
              virtual void disp() {cout << x << endl; }
};

class Derived: public Base {   int y;
  public:
    Derived(int val1 = 0, int val2 = 0) : Base(val1) { y = val2 + val1; }
    void operator++() { ++x; ++y;}
    void disp() { cout << "x=" << x << "   y=" << y << endl; }
};

int main ()
{
  Base *p;
  Derived d(3, 5);
  Base b = d;
  ++b; b.disp(); d.disp();
  ++d; d.disp();
```

4
x =3 y = 8
x = 4 y = 9
x = 5 y = 10
x = 5 y = 10

```
  p = &d;   ++(*p);   p->disp(); d.disp();

  return 0;
}
```

5、写出下列程序的执行结果

```
class   model {
  friend bool operator !(model m1)
    { return   m1.n   != m1.m ;    }
  private:
    int   n,   m;
  public:
    model(int t1, int t2) { n = t1 ; m = t2; }
    operator int () const { return n + m;}
};
void main()
{    model s (30, 40);
     cout << ( !s ? s + 10 :s – 10) << endl;
}
```

80

6、写出下列程序的执行结果

```
class CST
{
  friend ostream & operator<< (ostream &os, const CST &ob );
  friend CST operator+(const CST &op1, const CST &op2)
      { CST tmp;
        tmp.data = op1.data + op2.data;
        return tmp;
      }

  public:
    CST( )   { data = count++; cout << "constructing " << data << endl;}
    ~CST( )   {   count--; cout << "deconstructing " << data << endl;}
    static int count;
  private:
    int data;

};

ostream & operator<< (ostream &os, const CST &ob )
{ os << ob.data;
  return os;
```

```
}

int CST::count = 10;

int main()
{ CST   cs, *ptr1,*ptr2;
   cout << "CST::count = " << CST::count << endl;

   ptr1 = new CST; ptr2 = new CST;
   cs = *ptr2 + *ptr2;
   cout << *ptr2 << " + " << *ptr2 << " = " << cs << endl;

   delete ptr1; delete ptr2;
   cout << "CST::count = " << CST::count << endl;

   return 0;
}
```

```
Constructing 10
CST::count = 11
Constructing 11
Constructing 12
Constructing 13
Deconstructing 24
Deconstructing 24
12 + 12 = 24
Deconstructing 11
Deconstructing 12
CST::count = 10
Deconstructing 24
```

7、写出下列程序的执行结果

```
class CBase {
      public:
          CBase(int i)
             { m_data = i;
               cout << "Constructor of CBase. m_data=" << m_data <<endl;
             }
           virtual   ~CBase()
             { cout << "Destructor of CBase. m_data=" << m_data <<endl; }
      protected:
        int m_data;
   };

   class CDerived: public CBase
   {
      public:
          CDerived(int i) : m_data(i), CBase(i+10)
             { cout << "Constructor of CDerived. m_data = " << m_data << endl; }
          ~CDerived()
             { cout << "Destructor of CDerived. m_data = " << m_data <<endl; }

       private:
```

```
        int m_data;
};

int main()
{
    CBase   *p ;
    p = new CBase(10);
    delete p;
    p = new CDerived (10);
    delete p;

    return 0;
}
```

```
Constructor of CBase. m_data=10
Destructor of CBase. m_data=10
Constructor of CBase. m_data=20
Constructor of CDerived. m_data =10
Destructor of CDerived. m_data = 10
Destructor of CBase. m_data=20
```

8、写出下列程序的执行结果，并说明该程序有什么问题，应该如何修改程序？

```
class CBase {
  public:
    CBase(int i)
      { m_data = i;
        cout << "Constructor of CBase. m_data=" << m_data <<endl;
      }
     ~CBase()
      { cout << "Destructor of CBase. m_data=" << m_data <<endl; }
  protected:
    int m_data;
};

class CDerived: public CBase
{
  public:
    CDerived(const char *s): CBase(strlen(s))
      { m_data = new char[strlen(s) + 1];
        strcpy(m_data, s);
        cout << "Constructor of CDerived. m_data = " << m_data << endl;
      }
    ~CDerived()
      {   delete m_data;
        cout << "Destructor of CDerived. m_data = " << m_data <<endl;
      }
```

```
    private:
        char   *m_data;
};

int main()
{
    CBase   *p ;
    p = new CDerived ("abcd");
    delete p;

    return 0;
}
```

执行结果：
Constructor of CBase. m_data=4
Constructor of CDerived. m_data=abcd
Destructor of CBase. m_data=4
问题：程序会造成内存泄漏
解决方法：将基类的析构函数改为虚函数

三. 程序填空（每空 2 分，共 24 分）

1、下列程序的输出是：

5 5
7 12
0 12

请填空。

```
class CConst {
public:
    CConst(int d = __0__) : __len(d)__ {  size += d;  }
    void Print() __const__ {  cout << len << " " << size <<endl;  }
private:
    int len;
    static int size;
};

int  CConst::size = 0;
int main()
{   CConst c(5）;
    c.Print();
    CConst c2(7);
    c2. Print ();
    CConst c3;
    c3. Print ();
    return  0;
}
```

2、在下面横线处填上适当字句，完成类中成员函数的定义。

```
class A{
```

```
friend   bool   operator==( A a1,  A a2) // 判两数据成员指向的对象相同
  {  return  strcmp(a1.a, a2.a) == 0      ; }
public:
     A(char *aa = "")        //用 aa 初始化 a 所指向的动态对象
       {  a =  new char[strlen(aa) + 1]   ;  strcpy(a, aa);  }
     A( const  A &other     )  //拷贝构造函数，构造一个和参数完全相同的对象
       {  a = new char[strlen(other.a) + 1]   ;
          Strcpy(a, other.a)              ;
       }
     ~A(){  delete  a     ;   }  //释放动态存储空间
private :
     char * a;
};
```

四. 编程（共 16 分）

1、设计一个处理时间的类 Time。Time 类能保存一个时间。它提供的功能有：

前缀的++：将当前时间加 1 秒。

加法操作：将两个时间相加。例如，11 点 15 分 50 秒加 3 点 30 分 20 秒的结果是 14 点 46 分 10 秒。

以 HH:MM:SS 的格式输出对象保存的时间。如 15：20：45

以 HH:MM:SS AM/PM 的格式输出对象保存的时间。如 15：20：45 被输出为 3:20:45 PM

评分标准：

正确写出类定义：6 分

构造函数：　　2 分

Operator++函数　2 分

加法函数　　2 分

两个输出函数　　各 2 分

一。选择填空（每空1分，共10分）

1. 假定类AA有如下定义，

```
Class AA
{int data;
 public: AA(int a=0){ data = a;}
        int Getvalue( ) const { return data; }
 };
```

x为该类的一个对象，则访问x对象中数据成员data的格式为__B__。

A. x.data;　B. x.GetValue();　C. x->GetValue( );　D. x.data( ) ;

2. 假定AA为一个类，int a()为该类的一个成员函数，若该成员函数在类定义体外定义，则函数头为__C__。

A. AA::a();　B. int AA:a();　C. int AA::a();　D. AA::int a() ;

3. 假定AB为一个类，r2是AB类的对象，执行 “AB r1 = r2;”时将自动调用该类的__D__。

A. 带参构造函数　B. 无参构造函数　C. 赋值重载函数　D. 拷贝构造函数

4. 下列函数模板的定义中，合法的是__D__

A. template T abs（T x）{ return x < 0 ;}

B. template class < T >　Tabs（T x）{return x<0;}

C. template T < class T >　abs（T x）{return x<0;}

D. template< class T >　bool Tabs（T x）　{ return x < 0; }

5. 假定要对类X定义加号操作符重载成员函数，实现两个X类对象的加法，并返回相加后的结果，则该成员函数的声明语句为 __B__

A. X operator+(const X & a , const X & b);　B. X operator+(const X & a) const;

C. operator+(X　a);　D. X　& operator+( );

6. 假定AB为一个类，则执行 “AB a(2), b[3], *p[4];”语句时共调用该类构造函数的次数为__C__。

A. 3　B. 5　C. 4　D. 9

7. 对友元（friend）不正确的描述是：__C__。

A. 友元关系既不对称也不传递。

B. 友元声明可以出现在private部分，也可以出现在public部分。

C. 类的友元函数必须在类的作用域以外被定义。

D. 整个类都可以声明为另一个类的友元。

8. 下列运算符中不能重载成友元函数的是__B__。

A +　B =　C *　D !=

9. 若已知 char str[10]; 有语句 cin>>str;

当输入为：This is a program，所得到的结果是 str=___C___。

A。This is a program　　B。T　　C。 This　　D This is a

10。关于纯虚函数和抽象类的描述中，错误的是___C___。

A。纯虚函数是一种特殊的虚函数，它没有具体的实现。

B。抽象类是指具有纯虚函数的类。

C。一个基类说明中有纯虚函数，该基类的派生类不再是抽象类。

D。抽象类只能作为基类来使用，其纯虚函数的实现由派生类给出。

二。程序理解（每题 4 分，共 40 分）

1。写出下列程序的执行结果

```
#include <iostream.h>
#include <iostream.h>
class   Point{
 private:
    int   x, y;

 public:
    Point(int a=1,int b=2)
      { x = a; y = b;
       cout << "Point(" << x << "," << y << ")" << endl;
      }
  ~Point() { cout << "Point[" << x << "," << y << "]" << endl; }
 };

 Point Point0(0, 0);
 int main( )
 { Point   origin;
  { Point point1(-5, 7); }
  Point point2(3, 4);
  return 0;
 }
```

答案：

```
Point(0, 0)
Point(1, 2)
Point(-5, 7)
Point[-5, 7]
Point(3, 4)
Point[3, 4]
Point[1, 2]
Point[0, 0]
```

2. 下列程序运行后的输出结果

```
#include<iostream.h>
class A{
    public:
        A() { x = 1; y = 1; cout << "Default Constructor!" << endl; }
        A(int xx, int yy) : x(xx), y(yy)
          {cout << "2 parameters constructor!" << "x=" << x << " " << "y=" << y << endl;}
        A(int number)
            {x = number; y = 0;
             cout << "1 parameter constructor!" << " " << "x=" << x << " "
                  << "y=" << y << endl;}
        ~A() {cout << "Destrucor!" <<   "x=" << x << " " << "y=" << y << endl;}
     private:
        int x, y;
};

int main()
{ A *p=new A[2];
A   a1(10, 20);
A *p1=new A(4);

delete [] p;
delete p1;
return 0;
}
```

答案：

```
Default Constructor!
Default Constructor!
2 parameters constructor! x = 10 y = 20
1 parameter constructor! x = 4 y = 0
Destrucor! x = 1 y = 1
Destrucor! x = 1 y = 1
Destrucor! x = 4 y = 0
Destrucor! x = 10 y = 20
```

3. 写出下列程序运行后的输出结果

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

class CST
{ public:
```

```
    CST(int v1=0){ data=v1; count++; }
    ~CST( ){   count--; }
    CST &setData( int =0) ;
    void print() const { cout << "data = " << data << endl;}
    static int count;
  private:
    int data;
};

CST &CST::setData( int v )
{ data = v;
  return *this;
}

int CST::count = 10;

int main()
{ CST cs[2],*ptr1,*ptr2;

  cout << "CST::count = " << CST::count << endl;

  ptr1 = new CST(4);
  ptr1->setData(5);
  ptr1->setData() = 10;
  ptr1->print();

  cout << "CST::count = " << CST::count << endl;
  delete ptr1;
  cout << "CST::count = " << CST::count << endl;

  ptr2 = cs;
  cout << "CST::count = " << CST::count << endl;

  return 0;
}
```

答案：

```
CST::count = 12
data = 10
CST::count = 13
CST::count = 12
CST::count = 12
```

4．写出下列程序运行后的输出结果

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

class CTyre
{ public:
    CTyre(int num=0){ cout << "A car needs " <<num <<" tyres."<<endl;}
    ~CTyre(){ cout << "Tyres are worn out"<<endl;}
  private:
    int tyre_number;
};

class CEngine
{ public:
    CEngine() { cout << "A car needs a strong engine."<<endl; }
    ~CEngine() { cout << "The Engine is out of work."<<endl;}
};

class CCar
{
  public:
    CCar():tyre(4){cout <<"This is my car "<< endl;}
private:
    CEngine engine;
    CTyre tyre;
};

int main()
{ CCar car;

  return 0;
}
```

答案：

A car needs a strong engine.
A car needs 4 tyres.
This is my car
Tyres are worn out
The Engine is out of work

5、写出以下程序的执行结果

```
#include<iostream.h>
#include <iomanip.h>
```

```
template <class T>
class Sample
{    T   n;
  public:
     Sample(T i){ n = i;}
     Sample operator*(int k);
     Sample operator++() {++n; return *this;}
     void disp(){cout << "n=" << setfill ( '$' ) << setw ( 10 ) << n << endl;}
};

template <class T>
Sample<T> Sample<T>::operator*(int k)
{ n = n * k;
  return *this;
}

int main()
{ Sample<int> s(20);
  s = s * 10;
  s.disp();
  Sample<double> f(10.5);
  f++;
  f.disp();
  return 0;
}
```

答案：

$$$$$$$200

$$$$$$11.5

6、写出运行下列程序的结果。

```
#include <iostream.h>
void testfun(int test)
{ if   (test < 0 )   throw test;
 if (test == 5)   throw "it is equal to 5";
 cout << test << '\t';
}

int main()
{ for (int i = -1; i < 10; ++ i)
      try {     testfun(i); }
          catch(int i)      { cout << "Except occurred: " << i << endl;  }
          catch(const char *s) {    cout << "Except occurred: " << s << endl; }
  return 0;
```

```
}
```

答案：

```
Except occurred: -1
0    1    2    3    4
Except occurred: it is equal to 5
6    7    8    9
```

7. 写出以下程序的执行结果

```
#include <iostream.h>
class Base
{ protected:
      int x;
  public:
      Base() { x = 0; }
      Base(int val) { x = val; }
      void operator++() { x++; }
};

class Derived:public Base
{ int y;
  public:
      Derived(){ y = 0; }
      Derived(int val1,int val2):Base(val1){ y = val2; }
      void operator--(){ x--; y--;}
      void disp()
         { cout << "x=" << x << "    y=" << y << endl; }
};
int main ()
{ Derived d(3, 5);
  d.disp();
  ++d;
  d.disp ();
  --d;
  d.disp();
}
```

答案：

```
x = 3    y = 5
x = 4    y = 5
x = 3    y = 4
```

8 写出以下程序的执行结果

```
#include <iostream>
```

```
#include <iomanip>
using namespace std;
int main()
{int n=100;
 cout<<setw(10)<<setfill('*');
 cout<< "decimal:" << n<<endl;
 cout<< "hexdecimal:" << hex << n << endl;
 cout<< "oct:" <<oct<< n<<endl;
 return 0;
}
```

大案：

**decimal:100

hexdecimal:64

oct:144

9。写出下列程序的运行结果

```
#include <iostream>
using namespace std;
class B
{ public:
    void f() { cout << "This is function f() in class B" << endl; }
    void virtual g() { cout << "This is function g() in class B" << endl; }
};
class D: public B
{ public:
    void f() { cout << "This is function f() in class D" << endl; }
    void virtual g() { cout << "This is function g() in class D" << endl; }
};

int main()
{   B b, *pb;
    D d;

    pb = &b;
    pb->f();    pb->g();
    d.f();    d.g();
    b = d;
    pb->f();    pb->g();
    pb = &d;
    pb->f();    pb->g();
```

```
    return 0;
}
```

答案：

```
This is function f() in class B
This is function g() in class B
This is function f() in class D
This is function g() in class D
This is function f() in class B
This is function g() in class B
This is function f() in class B
This is function g() in class D
```

10。写出下列程序的运行结果

```
#include<iostream>
using namespace std;
template<class W>
class h
{ public:
    W shows(W c){v = c, return v;}

  protected:
    W v;
};

template<class W, class T>
class s: public h<W>
{ public:
    T shows(T m){x = m; return x;}
 private:
    T x;
};

int main()
{s<int,double> d;
  cout << d.shows(3) << endl;
  cout << d.shows(4.7) << endl;
  return 0;
}
```

答案：3

4.7

三。程序填空（每空2分，共30分）

1.在下面横线处填上适当字句，完成类中成员函数的定义。

```
class A{
 public:
      A(int aa = 0) { a =  new int(aa) ;  }  //用 aa 初始化 a 所指向的动态对象
      A(  const A &  ) ;  //拷贝构造函数原型定义
      ~A(){  delete a ;   }//释放动态存储空间
 private
      int * a;
};
```

2.完成下列程序，使得输出为： 2 3 2

```
#include <iostream>
using namespace std;
class Counter{
    unsigned value;
public:  Counter(int v=0)  { value=v; }
         friend Counter operator++(Counter &);
         friend Counter operator++(Counter &, int);
         void display()const { cout << value << " "; }
};
Counter operator++(Counter &cc)
{ cc.value++;
  return cc;
}
Counter operator++(Counter&cc,int)
{   Counter t;
    t.value=cc.value++;
    return t;
}
int main()
{   Counter a(1), b(2), c;
    ++a;
    a.display();
    c=b++;
    b.display();
    c.display();
    return 0 ;
}
```

3。下面程序的作用是将文件 old 的内容复制到文件 new 中，在复制过程中，将其中的小写字母改成大写字母

```
#include<iostream.h>
```

```
#include<fstream.h>

void main()
{ char ch;
  ifstream file1(   "old"   );
  ofstream file2(   "new" );

  while( (ch = file1.get()) !=   EOF )
      {   if(ch>='a'&& ch<='z')   ch=ch-'a'+'A';
          file2.put(ch);
      }
   file1.close();
   file2.close();
}
```

4。point 类的定义如下

```
class point{
   int  x, y;
 public:
     point( int a = 0 ,  int b = 0 ){x = a; y = b ;}
      void show() {cout << " x=" << x << " , y=" << y << endl ;}
 } ;
```

其功能的测试程序为：

```
void main()
{   point p1(10,20); p1.show();
    point p2; p2.show();
    point p3(30); p3.show();
}
```

若 main()的运行结果为：

```
    x=10 , y=20
    x=0 , y=0
    x=30 , y=0
```

试将 point 类的定义与实现补充完整。

四。编程题(共 20 分)

1。（10 分）设计一个类 A，它具有重载的运算符＋和－，以及成员函数 show()，使得下列主函数的运算结果是：

x=5 y=20

x=1 y=10

主函数如下：

```
#include<iostream>
using namespace std;
void main()
{ A a1(3,15), a2(2, 5), a3, a4;
    a3=a1+a2;
    a4=a1-a2;
    a3.show();
    a4.show();
}
```

评分标准：类定义　　2 分
　　每个成员函数的实现　2 分* 4

2。（10 分）定义一个 shape 类记录任意形状的位置。在 shape 类的基础上派生一个 Rectangle 类和 Circle 类，在 Rectangle 类的基础上再派生一个 Square 类。必须保证每个类都有计算面积和周长的功能。保证下列程序的输出结果为：

```
  31.4
  (0, 0)
  12
  40
  (1, 2)
int main()
{ Circle   c1(0, 0 ,5);
  Rectangle r1(1, 2, 3, 4) ;
  Square   s1(1,2,10) ;

 cout << c1.circum() << endl ;
 cout << c1.position() << endl ;
 cout << r1.area() << endl ;
 cout << s1.circum() << endl;
 cout << s1.position() << endl;

  return 0;
}
```

评分标准：　正确实现继承　2 分
　　定义 Shape 为抽象类　1 分
　　Shape 类的定义　　1 分
　　每个类定义及实现　2 分*3

一．选择题：

1.若有以下说明:

int a[10]={1,2,3,4,5,6,7,8,9,10},*p=a;

则对数组元素地址的正确表示是__(D)_____

(A)&(a+1) (B)a++ (C)&p (D)a+1

2.派生类的对象对它的基类成员中什么是可访问的_____(A)_____。

A 公有继承的公有成员 B 公有继承的私有成员

C 公有继承的保护成员 D 私有继承的公有成员

3. 下面有关重载函数的说法中正确的是____(C)_____

A 重载函数必须具有不同的返回值类型； B 重载函数形参个数必须相同；

C 重载函数必须有不同的形参列表； D 重载函数名可以不同；

4. 对类的构造函数和析构函数描述正确的是___(A)_____

A. 构造函数可以重载，析构函数不能重载

B. 构造函数不能重载，析构函数可以重载

C. 构造函数可以重载，析构函数也能重载

D. 构造函数不能重载，析构函数也不能重载

5. C++支持两种多态，包括编译时多态和运行时多态，编译时多态和运行时多态分别通过__(A)_____来实现。

A. 重载和虚函数 B. 重载和重载

C. 虚函数和重载 D. 虚函数和虚函数

6. 类友元运算符 obj1+obj2 被编译器解释为 _____(D)_____

A. operator+(obj1,obj2) B. +(obj1,obj2)

C. obj2.operator+(obj1) D. obj1.operator+(obj2)

7. 关于纯虚函数和抽象类的描述中，错误的是__(C)__

(A) 抽象类只能作为基类使用，其纯虚函数的实现由派生类给出

(B）纯虚函数是一个特殊的虚函数，它没有具体的实现

(C）一个基类中说明有纯虚函数，该基类的派生类一定不再是抽象类。

(D）抽象类是指具有纯虚函数的类

二．.读程序写结果：

1.
```
void function(int b[], int size)
  {if (size>0)
      {
       function(&b[1], size-1);
       cout<<b[0];}
  }
   main()
    {
       int a[5]={10,20,30,40,50};
        function(a,5);   }
```

答案：50，40，30，20，10

```
2.#include <iostream.h>
class FirstLevel {
public:
     FirstLevel( int count = 12 ) { cout << "FirstLevel Constructor: " << count <<
endl; }
    virtual void print() const { cout << "FirstLevel print"    << endl; }
    ~FirstLevel() { cout << "FirstLevel Destructor" << endl; }
};
class SecondLevel : public FirstLevel {
public:
    SecondLevel( int count = 5 )   { cout << "SecondLevel Constructor: " << count
<< endl;}
    void print( ) const { cout << "SecondLevel print"    << endl; }
};
class ThirdLevel : public SecondLevel {
public:
    ThirdLevel( int count = 33 )   { cout << "ThirdLevel Constructor: " << count <<
endl;}
    void print( ) const { cout << "ThirdLevel print"    << endl; }
};
int main()
{ThirdLevel third(41);
     FirstLevel &first = third;
    first.print();
    return 0;
}
```

运行结果：

```
FirstLevel Constructor: 12
SecondLevel Constructor: 5
ThirdLevel Constructor: 41
ThirdLevel print
FirstLevel Destructor
```

```
3.#include <iostream.h>
template <class T>
class Base
{    public:
         void SetB(T val = 0) { Bitem = val; }
         virtual void Print() { cout << Bitem << endl; }
         T GetB() {return Bitem;}
    private:
         T Bitem;
```

```
};
template <class T1, class T2>
class Derived : public Base<T1>
{    public:
        void SetD(T1 val1 = 0, T2 val2=0) { Ditem = val2; SetB(val1);}
        void Print() { cout <<Ditem<<" "<<GetB()<<endl;}
        T2 GetD() { return Ditem;}
     private:
        T2 Ditem;
};
void main()
{    Derived<char*,double> d1;
     d1.SetD("My God!",57.5);
     Base<char*> b = d1;
     b.Print();
     d1.Print();

     Derived<char*,char*>d2;
     d2.SetD("You are your God!","Work hard!");
     d2.Print();

     Derived<char *, int>d3;
     d3.SetD("more more passed! ",60);
     Base<char*> *pb = &d3;
     pb->Print();
     d3.Print();
}
```

运行结果：

```
My God!
57.5 My God!
Work hard! You are your God!
60 more more passed!
60 more more passed!
```

4、

```
#include <iostream. h>
class A
{   private:
     int X, Y;
   public:
     A() { X = Y = 0;
          cout<< “Default Constructor called.” << endl;
```

```
    }
    A(int xx, int yy) {
        X = xx; Y = yy;
        cout<< "Constructor called." << endl;
    }
    ~A() {
        cout<< "Destructor called." << endl;
    }
};
void main()
{   A *p1 = new A;
    delete p1;
    p1 = new A(1,2);
    delete p1;
}
```

运行结果：

```
Default Constructor called.
Destructor called
Constructor called
Destructor called
```

三．填空：

1.假定在 C++程序中有如下说明：

```
int score[NUM_STUDENT];
```

要求将 score 数组中所有分数打印出来，并打印总分，打印格式如下：

```
 89
 99
100
 70
 60
 50
Total: 468
```

程序代码如下， 请填完整。

```
void main()
{   int i;
    int total;
    total = 0;
    for ( i = 0; i< NUM_STUDENT; i++ )
    {
        cout<<'\n'<< score[i]);
```

```
        total += score[i];
    }
    cout<<"Total:"<< total);
}
```

2.
```
#include<iostream>
using namespace std;
class MyClass
{private:
    enum { NUM = 100 }        or        static const int NUM = 100 ;
    int group[NUM];
    static int classID          ;
public:
    GetClassID() {return classID;}
};
int MyClass::classID = 12345;
void main()
{   MyClass a;
    cout<<a.GetClassID()<<endl;
}
```

四.编程：

编写类模板 Array 的程序，模板可以实例化任何元素类型的 Array 对象。

必须编写的成员函数有：构造函数，拷贝构造函数，重载等号（=）运算符，析构函数，重载下标（[]）运算符，获取数组大小（size）函数。

```
#include <iostream.h>
#include <assert.h>
template <class T>
class Array
{public:
    Array(int =10); //构造函数
    ~Array();       //析构函数
    Array(const Array &); //拷贝构造函数
     Array &operator=(const Array &); //重载等号运算符
    T& operator[](int) const;   //重载下标运算符
     int getSize() const; //获取数组大小
private:
    T *elems;
    int size;
};
template <class T>
Array<T>::Array(int initSize )
{    size = ( initSize >0 ) ? initSize : 10;
    elems = new T[size];
```

```
    assert( elems != 0 );
    for ( int i =0 ; i < size; i++ )
        elems[ i ] = 0;
}
template <class T>
Array<T>::Array(const Array &init) : size(init.size)
{   elems = new T[size];
    assert ( elems != 0 );
    for ( int i =0 ; i < size; i++ )
        elems[ i ] = init.elems[ i ];
}
template <class T>
Array<T>::~Array()
{ delete []elems;
}
template <class T>
Array<T> &Array<T>::operator=(const Array &right)
{   if( &right != this ){
        if ( size != right.size ) {
            delete []elems;
            size = right.size;
            elems = new T[ size ];
            assert( elems != 0 );
        }
        for ( int i =0 ; i < size; i++ )
            elems[ i ] = right.elems[ i ];
    }
    return *this;
}
template <class T>
T &Array<T>::operator [] ( int subscript ) const
{
    assert( 0 <= subscript && subscript < size);
    return elems[ subscript ];
}
template <class T>
int Array<T>::getSize() const
{return size;
}
```

上 海 交 通 大 学 试 卷（A卷）

（ 2007 至 2008 学年 第 二 学期 ）

班级号_____ 学号_____ 姓名 _____
课程名称 程序设计 （2） 成绩 _____

一. 选择填空（10分）

1. C++对C语言作了很多改进，下列描述中_____使得C语言发生了质变，从面向过程变成了面向对象。

A 增加了一些新的运算符；
B 允许函数重载，并允许设置缺省参数；
C 规定函数说明必须用原型；
D 引进了类和对象的概念；

2. 在一个单链表中，若S所指结点不是最后结点，在S之后插入T所指结点，则执行_____。

A. T→next = S; S→next = T;　　B. T→next = S→next; S→next = T;
C. T→next = S→next; S = T;　　D. S→next = T; T→next = S;

3. 下面叙述正确的是 _____ 。

A．抽象基类中所有的virtual函数都必须声明为纯virtual函数
B．使用基类指针引用一个派生类的对象是非常危险的
C．如果基类申明了一个纯virtual函数，派生类只有实现该函数才能成为具体类。
D．一个类中有virtual函数，该类就成为抽象类。

4. 已知类D是类B的公有派生类，并且 D *pd, d ; B *pb; 不符合赋值兼容规则的是_____。

A． pb = pd;
B． pd = pb;
C． pb = &d;
D． pd = &d;

5. 下面叙述正确的是_____。

A．“has-a”关于可以通过继承实现。
B．派生类不会继承基类的构造函数。
C．汽车类Car与车轮类SteeringWheel以及刹车装置类Brakes之间是“is-a”关系。
D．当销毁派生类对象时，析构函数的调用顺序和相应的构造函数的调用顺序相同。

我承诺，我将严格遵守考试纪律。

承诺人：_____

题号			
得分			
批阅人(流水阅卷教师签名处)			

6. C++支持两种多态，包括编译时多态和运行时多态，编译时多态和运行时多态分别通过_____来实现。

A 重载和虚函数　　B 重载和重载

C 虚函数和重载　　D 虚函数和虚函数

7. 关于纯虚函数和抽象类的描述中，错误的是_____

A 抽象类只能作为基类使用，其纯虚函数的实现由派生类给出

B 纯虚函数是一个特殊的虚函数，它没有具体的实现

C 一个基类中说明有纯虚函数，该基类的派生类一定不再是抽象类。

D 抽象类是指具有纯虚函数的类

8. 对类的构造函数和析构函数描述正确的是_____

A 构造函数可以重载，析构函数不能重载

B 构造函数不能重载，析构函数可以重载

C 构造函数可以重载，析构函数也能重载

D 构造函数不能重载，析构函数也不能重载

9. 在下面四项中，不是用来限制类中成员的访问权限的是_____。

A. private　B. public　C. protect　D. protected

10. 在类中说明的友元函数是_____。

A. 可以访问该类对象的私有成员的成员函数

B. 冠以关键词 friend 说明的一般函数

C. 没有 this 指针的成员函数

D. 与静态成员函数具有相同的功能

参考答案：　DBCBBACACA

二． 读程序，写结果（40 分）

1.（6 分）

```
class A
{    int x,y;
     public:
          A(int xx=0,int yy=0):x(xx),y(yy) {  cout << "A..." << x << " " << y<<endl;}
          ~A()    { cout << "~A..." << x << " " << y << endl; }
          void put()   { cout << x << "   " << y << endl; }
        };

A* fun(int x, int y) { A *p = new A(x,y);   return p; }

int main()
{   A *p1;
    p1 = fun(123,789);
    A a1;
    a1.put();
    delete p1;
    p1 = fun(333,999);
    cout << "程序结束: " << endl;
    return 0;
  }
```

2。（6分）

```
class coord
{ friend coord operator++(coord &op);
   private:
          int x,y;
   public:
         coord(int i=0,int j=0);
         void print();
};

coord::coord(int i,int j) { x=i; y=j; }
void coord::print() { cout << "    x:" << x << ",y:" << y << endl; }
coord operator++(coord &op)
       { ++op.x;
          ++op.y;
          return op;
        }

int main()
{    coord ob(5,10);
     ob.print();
     ++ob;
     ob.print();
     operator++(ob);
     ob.print();
     return 0;
}
```

3. （6 分）

```
class A
{ private:
      int a;
   public:
      A(int M) { a = M;}
      A() { a=0;}
      void seta(int x) { a = x; }
      void showA() { cout << "a=" << a << endl;}
   };

class B: public A {
   private:
      int b;
   public:
      void setB(int x, int y) { b = x; seta(y); }
      void showB(){ showA(); cout << "b=" << b << endl;}
   };
int main()
{ B obj;
   obj.seta(53);
   obj.showA();
   obj.setB(53,58);
   obj.showB();
   return 0;
}
```

4.（6分）

```
class C_A {
    public:
        C_A(char value) {
            data =  value;
            m_count++;
            cout << "Object " << data <<" constructor" <<endl;
            cout << "The number of objects is  " << m_count << endl;
          }
        ~C_A()
          { cout <<"Object " << data << " destructor" << endl;
            m_count--;
          }
    private:
        char  data;
        static int m_count;
};

void Func();

int C_A::m_count = 0;

int main()
{  C_A *pa = new C_A('a');

  Func();
  delete pa;
  Func();

  return 0;
}

void Func()
{  static C_A f('b');
  C_A g('c');
}
```

5.（6分）

```
class CAutoMobile{
public:
    CAutoMobile(const char *);
    virtual ~CAutoMobile() {delete [] m_Model;}
    char * getModel() const { return m_Model;}

    virtual double price() const = 0;
    virtual void display() const {cout << getModel() << "s' price is undefined" << endl;}

private:
    char *m_Model;
};

CAutoMobile::CAutoMobile(const char *model)
{   m_Model = new char[strlen(model) + 1];
    strcpy(m_Model,model);
}

class CCar:public CAutoMobile {
public:
    CCar(const char *model,double price = 0.0): CAutoMobile(model), m_Price(price) {};
    virtual double price() const { return m_Price;}
    virtual void display() const
      { cout << "Car:" << CAutoMobile::getModel() << "'s price is " << price() << endl; }
private:
    int m_Price;
};

void func1(const CAutoMobile   &a )
{   cout << "In func1() ";
    a.display();
}

void func2(const CAutoMobile   *a)
{   cout << "In func2() ";
    a->display();
}

int main()
{   CCar   m("Ferrari 430",300);
    cout<<"In main() ";
```

```
    m.display();

    func1(m);
    func2(&m);
    return 0;
}
```

6. （4分）

```
void func( int ) ;

int main()
{ int i = 49;
  try{
      while ( i > 0 )
          { func(i);
            cout << i << endl;
            i = i / 2 - 1;
          }
    }
    catch( int ex)  {cout <<   ex << endl; }

    return 0;
}

void func(int num )
{ if   ( ! (num % 3) )   throw 3;
 else if   ( !(num % 4) ) throw 5;
}
```

7.（6分）

```
class CDoor
{ public:
      CDoor() { cout << "Door is    up." << endl;}
      ~CDoor() { cout << "Door is down." << endl;}
};

class CWall
{ public:
      CWall()    { cout << "Wall is up." << endl;}
      ~CWall() { cout << "Wall is down." << endl;}
};

class CRoom
{ public:
        CRoom():m_Door(),m_Wall()    { cout << "Room is up." << endl;}
        ~CRoom() { cout << "Room is down" << endl;}
   private:
        CWall m_Wall;
        CDoor m_Door;
};

int main()
{     CRoom room;
      return 0;
}
```

三． 程序填空（30 分）

1．完成下列 string 类的定义。请填空。（6 分）

```
class string{
        char *str;
        public:
                string(char *s)
                {str = new char[strlen(_____)+1];
                _____;}
                ~string() {_____;}
                string &operator=(string &s)
                {if (_____) return *this;
                 _____;
                 str = new char[strlen(s.str)+1];
                 strcpy(str, s.str);
                 _____;}
};
```

2．（4 分）请填空使得输出为：4+3I

```
class complex {
        int real; // 实部
        int imag; // 虚部
  public:
        complex(int r=0, int i=0){ _____;  _____;}
        void show() { cout<<real<<"+"<<imag<<"I";}
        complex operator++() { real++; return * this; }
};

int main() {
  complex c(3,3);
  _____;
  _____;
  }
```

3. 下列程序通过把函数 Distance 定义为类 Point 的友元来实现计算两点之间距离的功能，请完成程序（6 分）

```
class Point
  { public:
      _____;
       Point(_____)   { X = a;Y = b;}
      void Print()
          {cout << "X=" << X< < endl;
           cout << "Y=" << Y << endl;}
    private:
      float X, Y;
  };

float Dis(Point &p, Point &q)
  { float result;
    _____;
    cout << result << endl;
    return result;
  }

int main()
{   Point p(1,1), q(10,10)
  Dis(p,q);
}
```

4. （8 分）在空中补上语句使得输出结果为

```
class data 5
class a
class data 5
class b
class c

class data{
      int x;
    public:
      data(int x){
          data::x=x;
          _____;
        }
    };
```

```
class a{
        data d1;
    public:
        a(int x):d1(x)
            {_____;  }
    };

class b:public a{
        data d2;
    public:
        b(int x):a(x),d2(x) {  _____;  }
    };

class c:public b{
    public:
        c(int x):b(x) {_____;  }
    };

int main()
  { c object(5);
    return 0;
  }
```

5. （6 分）补充函数 max 使得程序结果为

The max of i,j is: 56
The max of x1,x2 is: 56.56
The max of y1,y2 is: 673.365

```
_____
T   max (_____)
{ _____;}
int main()
{ int i = 10, j = 56;
  float x1 = 50.3, x2 = 56.56;
  double y1 = 673.365, y2 = 465.972;

  cout<<"the max of i,j is:"<<max(i,j)<<"\n";
  cout<<"the max of x1,x2 is:"<<max(x1,x2)<<"\n";
  cout<<"the max of y1,y2 is:"<<max(y1,y2)<<"\n";
  return 0;
}
```

四． 编程题(20 分)

1．编写一个出租车收费类，创建该类对象时告知路程，该对象能告知该收多少费。计费方式是起价 11 元，其中含 3 公里费用，以后每半公里 1 元。（10 分）

2. 设计一个小型公司的人员信息管理系统。该公司主要有四类人员：老板（Boss）、销售人员（Salesman）、兼职技术人员（technician）。人员基本信息包括：姓名（name）、编号(no)。具体要求：(10分)

（1）人员编号的起始值为8000，每增加一个人员信息，编号顺序加1。

（2）月薪计算方法：老板拿固定月薪10000元；销售人员月薪为底薪1000元加当月销售额的5%提成；兼职技术人员按每小时100元领取月薪，如果月工作时间超过30小时，超出部分按每小时150元计算。

（3）对每个成员变量必须完成相应的get和set函数。

（4）编写全局函数input完成输入一个人员信息的功能。

（5）编写全局函数display，显示所有员工的全部信息以及当月月薪总额。

（6）尽可能多地用到面向对象设计各种特性（虚函数、抽象类、const等）。

6.5