浙江大学2013－2014学年春夏学期
《面向对象程序设计》课程期末考试试卷

课程号：____________，开课学院：____________

考试试卷：A卷√、B卷（请在选定项上打√）

考试形式：闭√、开卷（请在选定项上打√），允许带____________入场

考试日期：2014年06月26日，考试时间：120分钟

诚信考试，沉着应考，杜绝违纪。

考生姓名：______________学号：______________所属院系：_____________

题序	一	二	三	四	五	六	七	八	总 分
得分									
评卷人									

1. Write the output of the code below（30%）

（每题3分，主要部分对的酌情扣1-2分）

1）

```
class A{
public:
        A& opeator=(const A& r)
        {
            cout << "In A::operator=().";
        }
};
class B{
public:
        B& opeator=(const B& r)
        {
            cout << "In B::operator=().";
        }
};
class C{
private:
        A a;
        B b;
        int c;
public:
};

void main()
{
    C m,n;
    m = n;
}
```

答案:
In A::operator=().
In B::operator=().

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

class counter{
private:
     int value;
public:
     counter():value(0)
     {}
     counter& operator++();
     int operator++(int);
     void reset()
     {
          value = 0;
     }
     operator int() const
     {
          return value;
     }
};

counter& counter::operator++()
{
     if (5 == value)
          value = 0;
     else
```

```
		value += 1;

	return *this;
}

int counter::operator++(int)
{
	int t = value;
	if (5 == value)
		value = 0;
	else
		value += 1;

	return t;
}

void main()
{
	counter a;
	while (++a)
		cout << "*****\n";
	cout << 0+a << endl;
	while (a++)
		cout << "*****\n";
	cout << 0+a << endl;
}
```

答案:

```
*****
*****
*****
*****
*****
0
1
```

3）

```
class Obj {
	char c;
public:
	Obj(char cc){
		c = cc;
		cout << "Obj::Obj(char cc) for " << c << endl;
```

```
    }
    ~Obj() {
        cout << "Obj::~Obj() for " << c << endl;
    }
};
void f() {   static Obj b('b'); }
void g() {   Obj c('c'); }

Obj a('a');
int main()
{
 cout << "inside main()" << endl;
      f();
      g();
      f();
      g();
      cout << "leaving main()" << endl;
      return 0;
}
```

答案:

```
Obj::Obj(char cc) for a
inside main()
Obj::Obj(char cc) for b
Obj::Obj(char cc) for c
Obj::~Obj() for c
Obj::Obj(char cc) for c
Obj::~Obj() for c
leaving main()
Obj::~Obj() for b
Obj::~Obj() for a
```

4）

```
template <class T>
void print(const T &val){  cout << val; }

template <>
void print(const double &f_val){  cout << (int)f_val << endl ;}

void main()
{
    print("Today's temperature: ");
    print(26.3);
}
```

答案:
Today's temperature: 26

5）

```
int f(int a)
{
	return ++a;
}
int g(int& a)
{
	return ++a;
}

void main()
{
	int m = 0, n = 0;
	m += f(g(m));
	n += f(f(n));
	cout << "m=" << m << endl;
	cout << "n=" << n << endl;
}
```

答案:
m=3
n=2

6）

```
template <typename T>
class FF{
	T a1,a2,a3;
public:
	FF(T b1, T b2, T b3):a1(b1),a2(b2),a3(b3)
	{}
	T Sum() const
	{
		return a1 + a2 + a3;
	}
};
void main()
{
	FF<int> x(2,3,4),y(-2,-3,-4);
	cout << x.Sum() << "\t" << y.Sum() << endl;
}
```

答案:

9　 -9

7）

```
class A{
public:
	A() { cout << "A::A() called.\n";   }
	virtual ~A() { cout << "A::~A() called.\n";   }
};
class B:public A
{
public:
	B(int i)
	{
		cout   << "B::B() called.\n";
		buf = new char[i];
	}
	virtual ~B()
	{
		delete[] buf;
		cout   << "B::~B() called.\n";
	}
private:
	char *buf;
};

void fun(A* p)
{
	 delete p;
}

void main()
{
	A *p = new B(15);
	fun(p);
}
```

答案:

```
A::A() called.
B::B() called.
B::~B() called.
A::~A() called.
```

8）
```
class A
{
public:
    A() { cout << "A( )" << endl;}
    ~A() {cout << "~A()" << endl;}
};
class B : public A
{
public:
    B() { cout << "B( )" << endl;}
    ~B() {cout << "~B()" << endl;}

};
void main()
{
    A *ap = new B[2];
    delete ap;
}
```
答案:
```
A( )
B( )
A( )
B( )
~A()
```
9)
```
class A
{
public:
    virtual ~A(){}
};
class B : public A
{
};
void main()
{
    B *bp;
    B b;
    A a1;
    A &a2 = b;
    try{
        bp = dynamic_cast<B *>(&a1);
        if (bp)
            cout << "Dynamic_cast   (1) OK!"<<endl;
        else
            cout << "Dynamic_cast   (1) Fail!"<<endl;
        bp = dynamic_cast<B *>(&a2);
        if (bp)
            cout << "Dynamic_cast   (2) OK!"<<endl;
        else
            cout << "Dynamic_cast   (2) Fail!"<<endl;
        B &b1 = dynamic_cast<B &>(a1);
        cout << "Dynamic_cast   (3) OK!" <<endl;
    }
    catch(...){
        cout << "Dynamic_cast (3) Fail!"<<endl;
    }
}
```

答案:

```
Dynamic_cast   (1) Fail!
Dynamic_cast   (2) OK!
Dynamic_cast (3) Fail!
```

10)

```
class A
{
public:
     A(){cout << "A()" << endl;}
     A(const A&){cout << "A(const A&)"<<endl;}
     ~A(){cout << "~A()" << endl;}
     A &operator =(const A&){
          cout << "operator="<<endl;
          return *this;
     }
};
void main()
{
     A a1,a2;
     a2 = a1;
     A a3 = a1;
}
```

答案：

```
A()
A()
operator=
```

A(const A&)
~A()
~A()
~A()

2. Please correct the following programs（point out the errors and correct them）(15%) （每题3分）

1)

```
#include <iostream.h>
class A
{
      int m;
      static int k;
public:
      A():m(1111){}
      static int GetM()const{
            return m;
      }
      static int GetK()const{
            return k;
      }
};
int A::k = 555;
void main()
{
      A a;
      cout << a.GetM()<<endl;
      cout << a.GetK()<<endl;
}
```

答案：

```
class A
{
      int m;
      static int k;
public:
      A():m(1111){}
      ~~static~~ int GetM()const{     //去掉static
            return m;
      }
```

```
    static int GetK()~~const~~{      //去掉const
        return k;
    }
};
int A::k = 555;
void main()
{
    A a;
    cout << a.GetM()<<endl;
    cout << a.GetK()<<endl;
}
```

2)

```
#include <typeinfo.h>
#include <iostream.h>
class A
{
    int m_x;
public:
};
class B : public A
{
    int m_y;
public:
    B(int x = 0,int y = 0){ m_x = x; m_y = y; }
};
void main()
{
    A *ap=new B;
    cout<<typeid(*ap).name()<<endl;
}
```

答案：

```
#include <typeinfo.h>
#include <iostream.h>
class A
{
    ~~int m_x;~~
protected:
     int m_x;                    //m_x变量声明为protected
public:
   virtual ~A(){}               //增加虚析构函数
};
class B : public A
{
     int m_y;
public:
     B(int x = 0,int y = 0){ m_x = x; m_y = y; }
};
void main()
{
     A *ap=new B;
     std::cout<<typeid(*ap).name()<<std::endl;
    delete ap;          //增加delete ap
}
```

3)

```
#include <iostream.h>
class Rational
{
public:
      Rational(int numerator = 0,int denominator = 1){
                n = numerator,d = denominator;}
private:
      int n, d;     // numerator and denominator

      friend const Rational &operator*(const Rational& lhs, const
Rational& rhs);
     friend bool operator==(const Rational& lhs,   const Rational& rhs);
};

const Rational &operator*(const Rational& lhs,   const Rational& rhs)
{
      static Rational result;
      //multiply lhs by rhs and put the product inside result
      result.n = lhs.n * rhs.n;
      result.d = lhs.d * rhs.d;
      return result;
}
bool operator==(const Rational &lhs,const Rational &rhs)
{
     return lhs.n * rhs.d == rhs.n * lhs.d;
}
void main(){
     Rational a(1,2), b(3,5), c(2,1), d(1,7);
     if ((a * b) == (c * d))   {
         cout << "Equal" << endl;
     }
     else {
          cout << "Unequal" << endl;
     }
}
```

答案:

```
#include <iostream.h>
class Rational
{
public:
       Rational(int numerator = 0,int denominator = 1){
                  n = numerator,d = denominator;}
private:
       int n, d;     // numerator and denominator

       friend const Rational ~~&~~operator*(const Rational& lhs, const
Rational& rhs);     //去掉&
      friend bool operator==(const Rational& lhs,   const Rational& rhs);
};

const Rational ~~&~~operator*(const Rational& lhs,   const Rational& rhs)
{   //去掉&
       ~~static~~ Rational result;        //去掉static
       //multiply lhs by rhs and put the product inside result
       result.n = lhs.n * rhs.n;
       result.d = lhs.d * rhs.d;
       return result;
}
bool operator==(const Rational &lhs,const Rational &rhs)
{
      return lhs.n * rhs.d == rhs.n * lhs.d;
}
void main(){
      Rational a(1,2), b(3,5), c(2,1), d(1,7);
      if ((a * b) == (c * d))   {
          cout << "Equal" << endl;
      }
      else {
           cout << "Unequal" << endl;
      }
}
```

4)

```
class B {
public:
     virtual void f(){}
};
class D: public B {
public:
     virtual void f() const{}
};

void main()
{
    const B *bp = new D;
    bp->f();
}
```

答案：

```
class B {
public:
     virtual void f() const{}      //加上const
};
class D: public B {
public:
     virtual void f() const{}
};

void main()
{
    const B *bp = new D;
    bp->f();
     delete bp;      //增加delete bp
}
```

5)

```
class A
{
public:
	static int f1() const
	{
		return m_i;
	}
	static int f2() const
	{
		return m_s;
	}
	static int f3() const
	{
		return ++m_i;
	}
private:
	int m_i;
	static int m_s;
};
int A::m_s = 0
```

答案：

```
class A
{
public:
	~~static~~ int f1() const		//去掉static
	{
		return m_i;
	}
	static int f2() ~~const~~		//去掉static
	{
		return m_s;
	}
	~~static~~ int f3() ~~const~~		//去掉static和const
	{
		return ++m_i;
	}
private:
	int m_i;
	static int m_s;
};
int A::m_s = 0
```

3. Fill in the blanks（20%）（每空1分）

```
#include <iostream>
using namespace std;
class Base{
private:
	int a;
public:
	Base(int a=0) ________________
	{}
	virtual const char* what_am_i() const
	{
		return "Base\n";
	}
	________________ ~Base(){}
};
class Derived:public Base{
	char *p;
public:
	Derived(char *p) ________________
	{
		______________________________________________;
		______________________________________________;
	}
	Derived(const Derived& obj) ________________
	{
		______________________________________________;
		______________________________________________;
	}
	__________________________ what_am_i()_________
	{
		return "Derived\n";
	}
	Derived& operator=(const Derived & rhs)
	{
		if (________________)
		   return *this;

		___________________________;
		___________________________;
		______________________________________________;
		______________________________________________;
		__________________;
	}
	void stringIs() const
	{
		cout << p << endl;
	}

	~Derived()
	{
		__________________;
	}
};

void main()
{
	Base *p;
	p = new Derived("hello");
	Derived *q;
	q = __________________________________________;
	if (__________________)
		q->stringIs();
	cout << p->what_am_i();
	cout << (*p).what_am_i();
	__________________;
}
```

答案:

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

class Base{
private:
	int a;
public:
	Base(int a=0) ___:a(a)_________
	{}
	virtual const char* what_am_i() const
	{
		return "Base\n";
	}
	___virtual____ ~Base(){}
};
```

```
class Derived:public Base{
      char *p;
public:
      Derived(char *p) _:Base()_________
      {
            _this->p = new char[strlen(p)+1]__;
            _strcpy(this->p, p)__;
      }
      Derived(const Derived& obj) _:Base(obj)____
      {
            __p = new char[strlen(obj.p)+1]__;
            __strcpy(p, obj.p)__;
      }
      _virtual const char*_ what_am_i()__const__
      {
            return "Derived\n";
      }
      Derived& operator=(const Derived & rhs)
      {
            if (__this == &rhs___)
              return *this;

            _delete[] p_;
            __Base::operator=(rhs)____;
            __p = new  char[strlen(rhs.p)+1]_____;
            __strcpy(p, rhs.p)______;
            __return *this___________;
      }
      void stringIs() const
      {
            cout << p << endl;
      }
      ~Derived()
      {
            __delete[] p____;
      }
};

void main()
{
      Base *p;
      p = new Derived("hello");
```

```
        Derived *q;
        q = _dynamic_cast<Derived*>(p)___;
        if (__q__)
              q->stringIs();
        cout << p->what_am_i();
        cout << (*p).what_am_i();
        _delete p___;
}
```

4. Program Design（35%）

Given declaration of class Person as:

```
class Person {
public:
  Person(char* name);
  Person(const Person& r);
  virtual ~Person() {}
  char* getName() const { return name; }
  virtual void print() const;
  bool operator ==(const Person&) const;
private:
  char* name;
};
```

Your job is to design a simulation program for a clinic, in which there are doctors and patients. To be specific, the tasks you should do are:

1. Complete the code for member functions of class Person.
2. Design a class Doctor, which is derived from Person, and represents his/her specializing field as a string, and a registration fee rate as an integer. Overide the print function to print out the information.
3. Design a class Patient, which is also derived from Person, and has his/her social security number as a string. Overide the print function to print out the information.
4. Design a class Bill, in which there is one object of Doctor and one object of Patient as members. Design a print function for Bill to print out all the information it has.
5. Write a test program to create at least two doctors, two patients and two bills. Print information of all the bills.

答案：

1. Complete the code for member functions of class Person.（10分）

```
        Person::Person(char *nameIn)              //五个成员函数每个2分  {
              name = new char[strlen(nameIn)+1];
              strcpy(name, nameIn);
        }
        Person::Person(const Person &other) {
              name = new char[strlen(other.name)+1];
              strcpy(name, other.name);
        }
        void Person::print() const {
              cout << name << endl;
        }
        bool Person::operator==(const Person &other) const {
          return strcmp(this->name, other.name) == 0;
        }
        Person::~Person() {
              delete[] name;
        }
```

2. Design a class Doctor, which is derived from Person, and represents his/her specializing field as a string, and a registration fee rate as an integer. Overide the print function to print out the information. （7分）

```
class Doctor : public Person {          //类名、继承正确1分
public:
    Doctor(char *nameIn, char * fieldIn, int regFreeRateIn)//构造函数1分
        : Person(nameIn), regFreeRate(regFreeRateIn) {
        field = new char[strlen(fieldIn) + 1];
        strcpy(field, fieldIn);
    }
      virtual void print() {       //重载print函数2分
         Person::print();
        printf("specializing field: %s\n", field);
        printf("registration free rate: %d\n", regFreeRate);
      }
    ~Doctor()     //析构函数1分
    {
        delete[] field;
    }
private:
      int regFreeRate;       //reg成员变量1分
      char * field;          //field成员变量1分
};
```

3. Design a class Patient, which is also derived from Person, and has his/her social security number as a string. Overide the print function to print out the information. （7分）

```
class Patient : public Person {       //类名、继承正确1分
public:
Patient(char *nameIn, char *socialSecurityNumberIn) //构造函数1分
    :Person(nameIn)
{
socialSecurityNumber =
                  new char[strlen(socialSecurityNumberIn) + 1];
strcpy(socialSecurityNumber, socialSecurityNumberIn);
}
virtual void print() {       //重载print函数2分
    Person::print();
    printf("social security Number: %s\n",
socialSecurityNumber);
}
~Patient()     //析构函数1分
{
   delete[]   socialSecurityNumber;
}
 private:
  char *socialSecurityNumber;     //成员变量2分

};
```

4. Design a class Bill, in which there is one object of Doctor and one object of Patient as members. Design a print function for Bill to print out all the information it has. （8分）

```
class Bill {   //类名2分
public:
      Bill(Patient *patientIn, Doctor *doctorIn)   //构造函数2分
        : patient(patientIn),
          doctor(doctorIn) {

      }
      void print() {        //print函数2分
        printf("doctor information as below:\n");
        doctor->print();
        printf("patient information as below:\n");
        patient->print();
      }
     private:          //成员变量2分
      Patient *patient;
      Doctor *doctor;
};
```

5. Write a test program to create at least two doctors, two patients and two bills. Print information of all the bills. （5分）

```
 #include <stdio.h>
#include <string.h>
 int main() {
        Doctor doctor1("Bill", "field 1", 3);         //create doctor 1分
        Doctor doctor2("Fredman", "field 2", 4);
        doctor1.print();
        doctor2.print();

        Patient patient1("Elvis", "0134-443");     //create patient 1分
        Patient patient2("Adman", "0244-334");
        patient1.print();
        patient2.print();

        Bill bill1(&patient1, &doctor1);              //create bill 1分
        Bill bill2(&patient2, &doctor2);
        bill1.print();                                //打印信息2分
        bill2.print();
        return 0;
}
```