面向对象

15元

浙江大学2016－2017学年秋冬学期

《面向对象程序设计》课程期末考试试卷

课程号：211C0010，开课学院：计算机学院

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 无 入场

考试日期：2017 年 01 月 18 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									

FILL IN THE ANSWER SHEET. ANY WRITTEN BELOW WILL NOT BE COUNTED.

1. Write the output of the code below (assuming all the header files are taken care of) (30%)

1)

```
//head.h
#ifndef FUNCTION_TEMPLATE
#define FUNCTION_TEMPLATE
template <typename T>
inline T Max(T x,T y)
{
	cout << "In template function Max, ";
	return (x>y)?x:y;
}
inline int Max(int x, int y)
{
	cout << "In function Max, ";
	return (x>y)?x:y;
}
#endif
```

In function Max, 2
In template function Max, 2.0
~~3.0 求输.~~ 同1

```
//test.cpp
#include "function_template.h"
int main()
{
	cout << Max(1,2) << endl;
	cout << Max(1.0,2.0) << endl;
	cout << Max(1,2.0) << endl;
}
```

2)

```
class Base{
public:
	virtual int f1(char x) const { return (int)(x); }
	virtual int f2(int x) { return (2*x); }
	virtual int f3(int x) { return (3*x); }
};
class Derived : public Base{
public:
	virtual int f1(char x) { return (int)(-x); }
	virtual int f2(double x) { return (x/2); }
	virtual int f3(int x) { return (x/3); }
};
void print(Base& b)
{
	cout << b.f1('a') << "\t" << b.f2(97) << "\t" << b.f3(99) << endl;
}
int main()
{
	Base b;
	Derived d;
	print(b);
	print(d);
}
```

12都多态. 不构成继承

name hiding

97

~~96~~ 194 297

~~-96~~ 194 33.

97

3)

```
class A{
	int i;
public:
	A(int ii=0):i(ii) { cout << "call A(int ii=0).\n";    }
	A(const A& a) {
		i = a.i;
		cout << "call A(const A&).\n";
	}
	void print() const { cout << "A::i = " << i << endl; }
};
class B : public A{
	int i;
	A a;
public:
	B(int ii = 0) : i(ii) { cout << "call B(int ii=0).\n";      }
	B(const B& b) {
		i = b.i;
		cout << "call B(const B&).\n";
	}
	void print() const {
		A::print();
		a.print();
		cout << "B::i = " << i << endl;
```

call A(int ii=0) 基成员.
call A(int ii=0).
call B(int ii=0).
A::i = 0
A::i = 0
B::i = 2.

call A
call A.

call B(const B&).

A::i=0
A::i=0
B::i=2.

```
        }
};
int main()
{
        B b(2);
        b.print();
        B c(b);
        c.print();
}
```

4)

```
class Exception{
public:
        Exception(string name="none"):m_name(name)
        {
                cout << "Generating an exception object, name is "<<m_name<< endl;
        }
        Exception(const Exception& old_e)
        {
                m_name = string("ex_") + old_e.m_name;
                cout << "copy an exception object, name is "<<m_name<< endl;
        }
        virtual ~ Exception ()
        {
                cout << "destroy an exception object, name is " <<m_name<< endl;
        }
        string GetName() {return m_name;}
protected:
        string m_name;
};
class A{
public:
        A()
        {
                cout << "A()" << endl;
        }
        int f(int i)
        {
                if (i>=10) {
                        Exception ex_obj1("ex_obj1");
                        throw ex_obj1;
                }
                else
                        return i;
        }
        ~A()
        {
                cout << "~A()" << endl;
        }
};
int main()
{
        try
        {
                A a;
                a.f(10);
                A b;
                b.f(10);
        }
```


A()
Gene…is exobj1
catch exception.
destroy … exobj1
~A().

copy
destory

```
        catch(Exception& m)
        {
                cout<<"catch exception"<<endl;
        }
        catch(...)
        {
                cout<<"catch unknow exception"<<endl;
        }
}
```

5)

```
class A{
public:
  int t;
  A() {
    t = 1;
    cout<<"A()"<<t<<endl;
  }
};
class B : virtual A{
public:
  B() {
    t++;
    cout<<"B()"<<t<<endl;
  }
};
class C :  virtual B, virtual A{
public:
  C() {
    cout<<"C()"<<endl;
  }
};
int main()
{
  C c;
}
```

~~C#~~ A() 1
B() 2
C()

默认保证的继承

6)

```
class Parent {
  int i;
public:
  Parent(int ii) : i(ii) {
    cout << "Parent(int ii)\n";
  }
  Parent(const Parent& b) : i(b.i) {
    cout << "Parent(const Parent&)\n";
  }
  Parent() : i(0) { cout << "Parent()\n"; }
  friend ostream&
    operator<<(ostream& os, const Parent& b) {
    return os << "Parent: " << b.i << endl;
}};
class Member {
  int i;
public:
  Member(int ii) : i(ii) {
    cout << "Member(int ii)\n";
  }
```

```
    Member(const Member& m) : i(m.i) {
      cout << "Member(const Member&)\n";
    }
    friend ostream&
      operator<<(ostream& os, const Member& m) {
      return os << "Member: " << m.i << endl;
};
class Child : public Parent {
  int i;
  Member m;
public:
  Child(int ii) : Parent(ii), i(ii), m(ii) {
    cout << "Child(int ii)\n";
  }
  friend ostream&
    operator<<(ostream& os, const Child& c){
    return os << (Parent&)c << c.m
              << "Child: " << c.i << endl;
};
int main() {
  Child c(2);
  Child c2 = c;
  cout <<  c2;
}
```

Parent(int ii) ∠ Member(int ii)
Child (int ii)
Parent (const Parent&)
Member (const Member&)
~~Child()~~
Parent: 2
Member: 2
Child: 2.

2. Please correct the following programs（point out the errors and correct them）

(10%)

1)

```
class A{
      int i;
public:
    A(int ii):i(ii){}
};
class B: public A{
      char *p;
public:
    B(char *p)
    {
        p = new char[strlen(p)+1];
       strcpy(p, p);
    }
    ~B()
    {
       delete p;
    }
};
int main()
{
    B b("hello");
    A *p = new B("world!");
}
```

~~加一句：~~
~~A()空~~
~~A无默认构造函数~~.
virtual ~B()

见飞卢

777

2)

```
class Exception {};
class OneException : public Exception {};

void f(int index) throw()
{
    if ( index < 0 ) throw new OneException();
}
int main()
{
    int k;
    cin >> k;
    try {
        f(k);
    } catch (...) {
        cout << "caught ... " << endl;
    } catch (Exception) {
        cout << "caught Exception" << endl;
    } catch (OneException) {
        cout << "caught OneException" << endl;
}}
```

走示抛出异常.

不要在堆上构造

必须是最后一个

从小到大

3 2 1

3. Fill in the blanks（25%）Pay attention to the comments. No fill may also be an answer.

```
#include <iostream>
#include <cmath>
using namespace std;
#define PI 3.14159
class Shape {
private:
    int ID;
    static int counter;
public:
    Shape():ID(counter++) {}
    int objectID() const { return ID; }
    virtual void error() const ;
    virtual double area() = 0;
    static int getcounter() { return counter; }
};
____(1)_____
/* Default error handling function provided by base class Shape, to display default
code for error.*/
____(2)_____

class Ellipse: public Shape
{
private:
    int lax,sax;
    static int counter;
public:
    Ellipse(int l,int s): lax(l),sax(s)
    {
```

```
        if (lax!=sax) counter++;
    }
    ____(4)_____
    /* Ellipse class to handle errors */
    ____(5)_____
    static int getcounter() { return counter; }
};
____(6)_____

class Circle: public Ellipse
{
public:
    Circle(int r): ____(7)_____
    {
        ____(8)_____
    }
    static int getcounter() { return counter; }
    ____(9)_____
    /* The Circle class does not want to make any special behavior for the error */
    ____(10)_____
private:
    static int counter;
};
____(11)_____

class Rectangle: public Shape
{
protected:
    int width,length;
    static int counter;
public:
    Rectangle(int w,int l): ____(12)_____
    {
        ____(13)_____;
    }
    ____(14)_____
    /* Rectangle class to handle errors */
    ____(15)_____
    static int getcounter() { return counter; }
};
____(16)_____

class Square: public Rectangle
{
public:
    Square(int r): ____(17)_____
    {
        ____(18)_____;
    }
    ____(19)_____
    /* The Square class does not want to make any special behavior for the error */
    ____(20)_____
    static int getcounter() { return counter; }
private:
    static int counter;
};
____(21)_____;
class Triangle: public Shape
{
    int a,b,c;
    static int counter;
```

width(w), length(l)

Rectangle(r, r)

```
public:
    Triangle(int a, int b, int c): a(a),b(b),c(c)
    {
        counter++;
    }
    ____(23)____
   /* Rectangle class to handle errors */
    ____(24)____
    static int getcounter() { return counter; }
};
____(25)____;

int main ()
{
    Shape *list[6] = {
        new Ellipse(2,4), new Circle(3),
        new Rectangle(3,5), new Square(4),
        new Triangle(1,2,2),new Ellipse(1,3)};
    for (int i=0; i<6; i++) {
        cout << list[i]->area() << '\n';
        list[i]->error();
    }
    cout << Ellipse::getcounter() << endl;  //output: 2
    cout << Circle::getcounter() << endl;  //output: 1
    cout << Rectangle::getcounter() << endl;  //output: 1
    cout << Square::getcounter() << endl;   //output: 1
    cout << Triangle::getcounter() << endl;   //output: 1
}
```

4. Program Design（35%）

Design a matrix class template: Matrix(m × n), m is the number of rows of the matrix, n is the number of columns of the matrix. Support the following code:

```
//test.cpp
#include "matrix.h"
#include <iostream>
using namespace std;

int main()
{
    Matrix<int> m1(2,3),m2(2,3);
    cin >> m1;  //if at runtime enter: 1 2 3 4 5 6
    cin >> m2;  //if at runtime enter: 6 5 4 3 2 1
    Matrix<int> m = m1 + m2;
    m(1,2) = 0;
    cout << m << endl; //output is
                       //  7    7    7
                       //  7    7    0
}
```

Part of the code has been given:

```
//Matrix.h
#ifndef MATRIX_H
#define MATRIX_H

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

```
template <typename T>
class Matrix {
public:
friend ostream& operator<< <>(ostream& os, const Matrix<T>& n);
      friend istream& operator>> <>(istream& in, Matrix<T>& n);
//Please add the rest of the code
};
#endif
```