中山大学软件学院 2009 级软件工程专业(2009 春季学期)

《程序设计（II)》期末试题（A卷)

（考试形式：开卷　考试时间：2 小时)

警示

《中山大学授予学士学位工作细则》第六条

考试作弊不授予学士学位

方向：_____　姓名：_____　学号：_____

Section A: Multiple Choice (20 points)

For each of the following questions, choose only ONE of the provided multiple-choice: A, B, C, D corresponding to the best answer for them.

1. _____ includes the concepts of polymorphism and inheritance.
(A) object-like programming
(B) object-oriented programming
(C) generic programming
(D) none of these

2. Which of the following does the C++ compiler NOT examine, in order to select the proper overloaded?
(A) types and order of the arguments in the function call
(B) the number of arguments in the function call
(C) the return type of the function
(D) it examines all of the above

3. Given the following function template

```
template < class T >
T maximum( T value1, T value2 )
{
   if ( value1 > value2 )
      return value1;
   else
      return value2;
}
```

what would be returned by the following two function calls?

```
maximum( 2, 5 );
maximum( 2.3, 5.2 );
```

(A) 5, a type-mismatch error
(B) 5, 5.2

(C) 2, 2.3
(D) two error messages

4. The following program segment is most usually used to

```
#ifndef X
   C++ codes;
#endif
```

(A) avoid the codes to be included (evaluated) more than one time .
(B) avoid the codes to contain any error.
(C) make the codes be more readable.
(D) make the codes be more easily debugged.

5. Which of the following is NOT TRUE of a constructor and destructor of the same class?

(A) they both have same name aside from the tilde (~) character.
(B) they are both called once per object (in general).
(C) they both are able to accept default arguments.
(D) both are called automatically, even if not defined in the class.

6. `static` member functions:

(A) can use the this pointer.
(B) can only access other static member functions and static variables.
(C) cannot be called until their class is instantiated.
(D) are also called destructors

7. Which statement about operator overloading is FALSE?

(A) New operators can never be created.
(B) The precedence of an operator cannot be changed by overloading.
(C) Overloading cannot change how an operator works on built-in types.
(D) Certain overloaded operators can change the number of arguments they take.

8. Copy constructors must use call by reference because

(A) otherwise the constructor will only make a copy of a pointer to an object.
(B) otherwise the program will get trapped in infinite recursion
(C) the copy created using call-by-value has function scope.
(D) the pointer needs to know the address of the original data, not a temporary copy of it.

9. _____ relationships correspond to _____.

(A) is a, composition
(B) has a, composition
(C) with a, inheritance
(D) as a, inheritance

10. Polymorphism is implemented via

(A) member functions

(B) virtual functions

(C) inline functions

(D) non-virtual functions

Section B: Short Answer (40 points)

Briefly answer the questions according the requirements.

1. (6 points) Please write proper C++ statements in each of the following two blanks to complete the definitions of class `MyClass`, such that the running result of the program is 10.

```
#include<iostream>
using namespace std;
class MyClass
{
public:
    ____(1)____
    ____(2)____
private:
    int x;
};
int main()
{
    MyClass my(10);
    cout << my.GetNum() << endl;
    return 0;
}
```

2. (6 Points) Please write proper C++ statements in each of the following two blanks to complete the definitions of class `Derived`, such that the running result of the program is 0377.

```
#include<iostream>
#include<string>
using namespace std;
class Base
{
     int x;
 public:
    Base(int n = 0) : x(n) { cout << n; }
    int getX() const { return x; }
};
```

```
class Derived : public Base
{
     int y;
 public:
     Derived(int m, int n) : ___(1)_____
     { cout << n; }
    ___(2)_____
};

int main()
{
   Derived d1(3);
   Derived d2(5,7);
   return 0;
}
```

3. (10 Points) Please complete the definitions of the two `operator++()` functions in the class `Point` such that the results of the program are:

```
x=2,y=2
x=2,y=2
x=3,y=3
```

You MUST put the definitions outside the class body.

```
#include<iostream>
using namespace std;
class Point
{
    int x;
    int y;
public:
    Point(int x = 0, int y = 0):x(x), y(y) { }
    Point& operator++(); //member function
    friend Point operator++(Point &, int);
    void print() { cout << "x=" << x<< ",Y=" << y << endl; }

};


int main()
{
   Point p1(1,1);
   Point p2(2,2);
```

```
    Point p3;
    (++p1).print();
    p3 = p2++;
    p3.print();
    p2.print();
    return 0;
}
```

4. (12 points) A structure for a point and a class for a circle are defined as following specifications. And the definitions of `Circle` are incomplete.

```
const double PI = 3.14159;

struct Point {
    double x;
    double y;
};

class Circle {
    Point point;
    double radius;
public:
    Circle(Point, double);
    double getCircumference();
    double getArea();
    bool isInside(Point);
};
```

(1) (3 points) Complete the definition of constructor `Circle()`.
(2) (3 points) Complete the definition of member function `getCircumference()` that returns the circumference of the circle.
(3) (3 points) Complete the definition of member function `getArea()` that returns the area of the circle.
(4) (3 points) Complete the definition of member function `isInside()` that passes a point as the parameter and decides if the point is inside the circle or not. If the point is inside the circle, returns `true` and otherwise return `false`.

5. (6 points)
The following blank should contain a C++ code segment that prints out every element (actually it is `d` in struct `data`) of vector `V`. Write two sets of codes to implement the same function.

```
#include <iostream>
#include <vector>
```

```
using namespace std;

struct data
{
   int d;
};

int main(void)
{
   vector<data> V;
   int i;
   data dt;

   for (i = 0; i < 10; i++) {
      cin >> dt.d;
      V.push_back(dt);
   }

      (1)

   return 0;
}
```

Section C: Program Output Analysis (40 points, 5 points each)

Write the result after executing the following programs or program fragments.

1.

```
#include<iostream>
using namespace std;
class ObjectCounter
{
public:
   ObjectCounter() { ++count; }
   ~ObjectCounter() { --count; }
   static void total_count()
   {
      cout << count << "ObjectCounter(s)" << endl;
   }
private:
   static int count;
};
```

```
int ObjectCounter:: count = 0;

int main()
{
   ObjectCounter p1[10];
   ObjectCounter::total_count();
   return 0;
}
```

2.

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

class A {
    int a;
public:
    A(int aa = 0) { a=aa; }
    ~A() { cout << "Destructor A!" << a << endl;  }
};

class B : public A {
    int b;
public:
    B(int aa = 0, int bb = 0) : A(aa) { b = bb; }
    ~B() {cout << "Destructor B!" << b << endl; }
};

int main() {
    B x(5);
    B y(6, 7);
    return 0;
}
```

3.

```
#include<iostream>
using namespace std;
class A
{
public:
   int n;
};
class B : public A { };
class C : public A { };
class D : public B, public C
```

```
{
   int getn() { return B::n; }
};

int main()
{
   D d;
   d.B::n = 10;
   d.C::n = 20;
   cout << d.B::n << "," << d.C::n << endl;
   return 0;
}
```

4.

```
#include<iostream>
using namespace std;
class A
{
public:
   A(char *s) { cout << s << endl; }
   ~A(){};
};

class B : virtual public A
{
public:
   B(char *s1, char *s2) : A(s1)
   {
       cout << s2 << endl;
   }
};

class C : virtual public A {
public:
   C(char *s1, char *s2) : A(s1)
   {
       cout << s2 << endl;
   }
};

class D : public B , public C
{
public:
```

```
    D(char*s1,char*s2,char*s3,char*s4):
           B(s1,s2),C(s1,s3),A(s1)
    {
          cout << s4 << endl;
    }
};

int main()
{
    D* p = new D("classA", "classB", "classC", "classD");
    delete p;
    return 0;
}
```

5.

```
#include<iostream>
using namespace std;
class GA
{
public:
   virtual int f() { return 1; }
};

class GB : public GA
{
public:
  virtual int f() { return 2; }
};

void show(GA g) { cout << g.f(); }
void display(GA &g) { cout << g.f(); }

int main()
{
   GA a;
   show(a);
   display(a);

   GB b;
   show(b);
   display(b);
   return 0;
}
```

6.

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

class MyClass
{
public:
   MyClass(int i = 0) { cout << i; }
    MyClass(const MyClass &x) { cout << 2; }
    MyClass& operator=(const MyClass &x)
    { cout << 3; return *this; }
    ~MyClass() { cout << 4; }
};

int main()
{
    MyClass obj1(1);
   MyClass obj2(2);
   MyClass obj3 = obj1;

    return 0;
}
```

7.

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

int main()
{
   list<int> myList;
   list<int>::iterator it;
   int i;

   for (i = 0; i < 10; i++)
      myList.insert(myList.begin(), i+1);

   for (it = myList.begin(); it != myList.end(); it++)
   {
      cout << *it << " ";
   }

   cout << endl;
   return 0;
```

```
}
```

8.

```
#include <iostream>
#include <cstring>
using namespace std;
class Pet
{
   char name[10];
public:
   Pet(char*name) { strcpy(this->name,name); }
   const char *getName() const { return name; }
   virtual void call() const = 0;
};

class Dog: public Pet
{
public:
   Dog(char *name) : Pet(name) { }
   void call() const { cout << "The dog is calling"; }
};

class Cat: public Pet
{
public:
   Cat(char *name) : Pet(name) { }
   void call() const { cout << "The cat is calling"; }
};

int main()
{
   Pet *pet1 = new Dog(" Hark "), *pet2 = new Cat(" Jimmy ");

   cout << pet1->getName(); pet1->call(); cout<<endl;
   cout << pet2->getName(); pet2->call(); cout<<endl;
   delete pet1;
   delete pet2;
   return 0;
}
```