計算機程式 期末考 Jan. 10, 2001

1. True or False Problems (20%)

(a) Data members of a class must be initialized in the class definition.

(b) Constructors of a class can be overloaded and can specify default arguments.

(c) A class name is also a type name and can be used to declare objects of that class.

(d) Any C++ operator can be overloaded.

(e) It's possible to create new operators as well as overloading existing operators in C++.

(f) Exceptions may occur in `try` blocks and will be handled in `catch` blocks.

(g) C++ views each file as a sequential stream of bytes.

(h) STL (standard template library) avoids virtual functions in favor of using generic programming with templates to achieve better execution-time performance.

(i) Using STL can save considerable time and effort due to the concept of program reuse, and result in higher quality programs.

(j) A base-class object can also be viewed as an object of its derived classes.

2. (10%) Find the error(s) in the following and explain how to correct it:

```
class Eclass {
public:
    Eclass( int x =10) { data = x;}
    Eclass();
    int getIncData() const { return data++;}
    static int getCount()
     {
        cout << "Data is " << data << endl;
        return count;
     }
private:
    int data;
    static int count;
}
```

3. (10%) Explain the difference between declaration and definition. Use member functions m1 and m2 as examples to explain where the two member functions are declared and where are defined, respectively.

```
#include <iostream>
using namespace std;
class B {
 public:
    B(int i=100, int j=101) { x=i; y=j;}
    virtual int m1() {x++; cout << " x= " << x << endl; return x;}
    virtual int m2();
 private:
    int x;
    int y;
```

```
};
class D: public B {
 public:
    D(int i) {z=i; }
    virtual int m1() {z=1; cout << " z= " << z << endl; return z;}
    virtual int m2() {z=2; cout << " z= " << z << endl; return z;}
 private:
    int z;
};
int B::m2() {
    y++; cout << " y= " << y << endl; return y;
}
int main() {
 B *bPtr = new D(10);
 bPtr -> m1();
 bPtr -> m2();
 bPtr = new B;
 bPtr -> m1();
 bPtr -> m2();
 return 0;
}
```

4. (10%) What are the outputs of the program in Problem 3?
5. (10%) Single choice problems.
   (a) Why don't we recommend the frequent use of protected member data? (i) it is not efficient (ii) it breaks the concept of encapsulation (iii) it cannot be inherited.
   (b) Why do we usually overload the insertion operator << as a friend function not as a member function? (i) insertion operator << cannot be overloaded as member function due to the syntax (ii) we want preserve the usual calling convention of the operator, such as cout << object (iii) insertion operator << cannot be overloaded as member function due to the run-time constraints.
   (c) Which statement is false? (i) Constructors can be declared as a virtual function. (ii) If a class has virtual functions, providing a virtual destructor even of one is not required for the class is a good programming practice. (iii) A class can inherit interface and/or implementation from a base class.
   (d) Which statement is false? (i) Referring to the `this` pointer within a static member function is a syntax error. (ii) Member initializers must be provided in the constructor of a class when that call contains const data members. (iii) A member function should declare the `this` pointer explicitly before using it.
6. (10%) Explain and correct the error in the following code.

```
class B {
   private:
      int x;
};
class D : public B {
   public:
      void f() { y = x + 1;}
```

```
    private:
        int y;
};
```

7. (10%) Explain and correct the error in the following code.

```
class B {
    public:
        B(int a) {x = a; y = 100;}
        B(int a, int b) {x = a; y = b;}
    private:
        int x;
        int y;
};
class D : public B {
    public:
        D(int n) { z = n;}
    private:
        int z;
};
```

8. (10%) What are the outputs generated by the following program?

```
#include <iostream>
using namespace std;
class B {
    public:
        B() { cout << " B's constructor" << endl;}
        ~B() { cout << " B's destructor" << endl;}
};
class C {
    public:
        C() { cout << " C's constructor" << endl;}
        ~C() { cout << " C's destructor" << endl;}
};
class D: public B {
    public:
        D() { cout << " D's constructor" << endl;}
        ~D() { cout << " D's destructor" << endl;}
};
int main(){
    B b;
    {C c;}
    D d;
    return 0;
}
```

9. (10%) Consider the following class definition and some of its member function's definition.

(a) Which is the default constructor and which is the copy constructor?

(b) Find the error(s) in the following and explain how to correct it:

```
class Array {
   friend ostream &operator<<( ostream &, const Array & );
   friend istream &operator>>( istream &, Array & );
public:
   Array( int = 10 );
   Array( const Array);
   ~Array();
   int getSize() const;
   const Array &operator=( const Array & );
   bool operator==( const Array & ) const;
   bool operator!=( const Array &right ) const
      { return ! ( *this == right ); }

   int &operator[]( int );
   const int &operator[]( int ) const;
   static int getArrayCount();

private:
   int size;
   int *ptr;
   static int arrayCount;
};

Array::Array( const Array init ) : size( init.size )
{
   ptr = new int[ size ];
   assert( ptr != 0 );
   ++arrayCount;

   for ( int i = 0; i < size; i++ )
      ptr[ i ] = init.ptr[ i ];
}

Array::~Array()
{
   delete  ptr;
   --arrayCount;
}
```