

# ECE 30862 Fall 2012, Second Exam

**DO NOT START WORKING ON THIS UNTIL TOLD TO DO SO. LEAVE IT ON THE DESK.**

You have until 12:20 to take this exam.

Your exam should have 16 pages total (including this cover sheet). *Please let Prof. Midkiff know immediately if it does not.* Each problem is worth 7 points unless noted otherwise.

This exam is open book, open notes, but no electronics. If you have a question, please ask for clarification. If the question is not resolved, state on the test whatever assumptions you need to make to answer the question, and answer it under those assumptions. *Check the front board occasionally for corrections.*

**Name:**

**Student ID:**

0.1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.18.19.20

## 1 C++ Question:

```
class B {
public:
    B() { }
    ~B() { }

    virtual void print() {
        cout << "B::print( ) ";
    }

    virtual void print(int i) {
        cout << "B::print(int)" << endl;
    }
};

class D : public B {
public:
    D() { }
    ~D() { }

    virtual void print() {
        cout << "D::print( ) ";
    }

    virtual void print(int i) {
        cout << "D::print(int)" << endl;
    }
};

int main(int argc, char * argv[]) {

    D *d = new D( );
    B *b = (B*) d;;

    b->print( );
    d->print(4);

    return 0;
}
```

What is printed:

- a: D::print( ) D::print(int)
- b: B::print( ) D::print(int)
- c: D::print( ) B::print(int)
- d: B::print( ) B::print(int)
- e: Compiler error because it is ambiguous which B and/or D is to be called

## 2 C++ Question:

```
class B {
public:
    B() { }
    ~B() { }

    void print() {
        cout << "B::print( ) ";
    }

    void print(int i) {
        cout << "B::print(int)" << endl;
    }
};

class D : public B {
public:
    D() { }
    ~D() { }

    void print() {
        cout << "D::print( ) ";
    }

    void print(int i) {
        cout << "D::print(int)" << endl;
    }
};

int main(int argc, char * argv[]) {

    D *d = new D( );
    B *b = (B*) d;;

    b->print( );
    d->print(4);

    return 0;
}
```

What is printed:

- a: D::print( ) D::print(int)
- b: B::print( ) D::print(int)
- c: D::print( ) B::print(int)
- d: B::print( ) B::print(int)
- e: Compiler error because it is ambiguous which B and/or D is to be called

### 3 C++ Question:

```
class B {
public:
    B() { }
    ~B() { }

    virtual void print() {
        cout << "B::print( ) ";
    }

    virtual void print(int i) {
        cout << "B::print(int)" << endl;
    }
};

class D : public B {
public:
    D() { }
    ~D() { }

    virtual void print() {
        cout << "D::print( ) ";
    }

    virtual void print(int i) {
        cout << "D::print(int)" << endl;
    }
};

int main(int argc, char * argv[]) {

    D *d = new D( );
    B *b = (B*) d;

    ((B) (*b)).print( );
    ((B) *d).print(4);

    return 0;
}
```

What is printed:

- a: D::print( ) D::print(int)
- b: B::print( ) D::print(int)
- c: B::print( ) B::print(int)
- d: D::print( ) B::print(int)
- e: Compiler error because it is ambiguous which B and/or D is to be called

## 4 Java Question:

```
class B {  
  
    public B( ) { }  
    public void print( ) {System.out.print("B::print( ) ");}  
    public void print(int i) {System.out.println("B::print(int)");}  
}  
  
class D extends B {  
  
    public D( ) { }  
    public void print( ) {System.out.print("D::print( ) ");}  
    public void print(int i) {System.out.println("D::print(int)");}  
}  
  
class Test {  
  
    public static void main(String args[]) {  
  
        D d = new D( );  
        ((B) d).print( );  
        d.print(4);  
    }  
}
```

What is printed:

- a: B::print( ) B::print(int)
- b: B::print( ) D::print(int)
- c: D::print( ) B::print(int)
- d: D::print( ) D::print(int)
- e: Compiler error because it is ambiguous which B and/or D is to be called

## 5 C++ Question:

```
void f(int i) {
    if (i < 0) {
        throw 29;
    } else {
        cout << ">= 0 ";
    }
}

void g( ) {
    for (int i = 0; i > -2; i--) {
        try {
            f(i);
            cout << i << " ";
        } catch (int i) {
            cout << "exception caught ";
        }
    }
    cout << "the end." << endl;
}

int main( ) {
    g( );
}
```

What is printed?

- a. >= 0 0 exception
- b. the end.
- c. the program terminates when the exception is thrown
- d. there is an compile-time error when throwing an exception that is not an object.
- e. >= 0 0 exception caught the end.

## 6 Java Question:

```
class E extends Exception {  
  
    public E( ) { }  
    public void print( ) {System.out.print("Bad things happened ");}  
}  
  
class Test {  
  
    private static void f(int i) throws E {  
        if (i > 0) throw new E( );  
    }  
  
    public static void main(String args[]) {  
  
        for (int i = 1; i > -1; i--) {  
            try {  
                f(i);  
                System.out.print(i+" ");  
            } catch (E e) {  
                e.print( );  
            }  
        }  
        System.out.println("terminating program");  
    }  
}
```

What is printed?

- a. Bad things happened 0 terminating program
- b. Bad things happened
- c. Bad things happened terminating program
- d. The program ends with an error before printing anything
- e. The program has a compile error because class Test does not extend the exception it catches.

## 7 C++ Question:

```
void f(int* p) {
    *p = 58;
    p = 0;
}

int main( ) {
    int i = 29;
    int* p = &i;
    cout << p << " ";
    cout << *p << " ";
    f(p);
    cout << p << " ";
    cout << *p << "x" << endl;
    return 0;
}
```

What is printed if “p” has the value “0x7fff61d3cb04”??

- a. 0x7fff61d3cb04 29 0x7fff61d3cb04 29x
- b. 0x7fff61d3cb04 29 0x7fff61d3cb04 58x
- c. 0x7fff6ad8eb04 29 0x000000000000
- d. The program has a null pointer error in the statement “cout << \*p << ”x” << endl;”
- e. None of the above.

## 8 Java Question:

```
class B {
    public int i;
    public B( ) {i = 29;}
}

class Test {

    public static void foo(B bb) {
        bb.i = 58;
        bb = null;
    }

    public static void main(String args[]) {

        B b = new B( );
        System.out.print(b.i+" ");
        foo(b);
        System.out.print(b.i+" ");
    }
}
```

What is printed?

- a. "29" is printed, followed by a null pointer error at the second "System.out.println" in main.
- a. 29 29
- b. 29 58
- c. None of the above.

## 9 C++ Question:

```
class Shape {
public:
    virtual double area( );
    virtual double circumference() = 0;
};

int main( ) {
    Shape s( );
}
```

What is the best answer?

- a. This is a legal program.
- b. There is an error since you cannot instantiate objects from an abstract class.
- c. `circumference` is an abstract function, `area` is not.
- d. b and c are both correct.
- e. The function `circumstance` will return 0 if not overridden with another function.

## 10 Java Question:

```
abstract class Shape {
    int i;
    abstract protected double area( );
    abstract protected double circumference( );
}
```

```
class Test {

    public static void main(String args[]) {
        Shape s = new Shape( );
    }
}
```

What is printed?

- a. This is a legal program as long as the methods `area` and `circumference` are not called.
- b. This is a legal program.
- c. There is an error because `Test` must extend `Shape` before creating a shape object.
- d. There is an error because an abstract class cannot declare data members.
- e. There is an error since you cannot instantiate objects from an abstract class.

## 11 Java Question:

```
interface Shape {
    int i = 0;
    public double area( );
    public double circumference( );
}
```

```
interface Color {
    int i = 0;
    public int red( );
    public int green( );
    public int blue( );
}
```

```
class Test implements Shape, Color {
// code for class Test goes here
}
```

Which answer is most correct?

- a. Class `Test` must define methods `area`, `circumference`, `red`, `green` and `blue`.
- b. Class `Test` must define methods all methods that it will use or that classes inheriting from `Test` will use.
- c. This program is inherently illegal because Java disallows multiple inheritance.
- d. This program is inherently illegal because the variable `i` is declared in both the `Shape` and `Color` interfaces.
- e. both c and d are correct.

## 12 Java Question:

```
interface Shape {
    int i = 0;
    public double area( );
    public double circumference( );
}

interface Color {
    int i = 0;
    public int red( );
    public int green( );
    public int blue( );
}

class Test implements Shape, Color {
    public double area( ) {return 0.0;}
    public double circumference( ) {return 0.0;}
    public int red( ) {return 0;}
    public int green( ) {return 0;}
    public int blue( ) {return i;}
}
```

Which answer is most correct?

- a. The program is legal because all of the methods in `Shape` and `Color` are implemented.
- b. The program is illegal because it is ambiguous which `i` is being used in the definition of `blue` in class `Test`.
- c. The `i` used in the definition of `blue` in class `Test` is the `i` defined in `Color`.
- d. This program is inherently illegal because Java disallows multiple inheritance.

### 13 Language neutral question:

You would like to have a container that efficiently implements inserts and deletes from the front, the end and the middle of the container. You should use

- a. a vector
- b. a queue
- c. a list
- d. a or b.
- e. all are equally good.

## 14 C++ Question:

```
class MyComplex {
private:
    double re, im;
public:
    MyComplex(double r, double i) : re(r), im(i) { }
    // MyComplex(const MyComplex& orig) { re = orig.re; im = orig.im; }

    MyComplex operator-( const MyComplex& arg) {
        return MyComplex(re-arg.re, im-arg.im);
    }

    MyComplex operator-( ) {
        return MyComplex(-re, -im);
    }

    friend ostream& operator<< (ostream& os, const MyComplex& arg);
};

ostream& operator<< (ostream& os, const MyComplex& arg) {
    os << "(" << arg.re << ", " << arg.im << ")" << endl;
    return os;
}

int main( ) {
    MyComplex first(3,4);
    cout << first - -first << endl;
    return 0;
}
```

Which answer is most correct?

- a. `ostream& operator<<` is a **friend** of class `MyComplex` because it accepts a `MyComplex` as an argument, and is not a member of the `MyComplex` class.
- b. There is no need in this program for the `ostream& operator<<` to be a **friend** of class `MyComplex`, and could have been a member function of the class `MyComplex`.
- c. If operator overloading is used, there must be at least one **friend** class for every class that implements overloaded operators.
- d. `ostream& operator<<` is a **friend** of class `MyComplex` so that it can access private fields in `MyComplex`.

## 15 C++ Question:

```
class MyComplex {
public:
    double re, im;
    MyComplex(double r, double i) : re(r), im(i) { }
    MyComplex operator-(const MyComplex& arg); // FUNCTION 1};
};

MyComplex MyComplex::operator-( const MyComplex& arg) {
    double d1 = re - arg.re;
    double d2 = im - arg.im;
    return MyComplex(d1, d2);
}

MyComplex operator-(const MyComplex& arg) {
    double d1 = -arg.re;
    double d2 = -arg.im;
    return MyComplex(d1, d2);
}

int main( ) {
    MyComplex first(3,4);
    MyComplex second(2,9);

    first - second;
    return 0;
}
```

Which answer is most correct?

- a. “MyComplex MyComplex::operator-” and “MyComplex operator-” are both unary operators
- b. “MyComplex MyComplex::operator-” is a binary operator and “MyComplex operator-” is a unary operators
- c. “MyComplex MyComplex::operator-” is a unary operator and “MyComplex operator-” is a binary operators
- d. “first - second” calls “MyComplex MyComplex::operator-” and arg.re refers to seconds’s re field.
- e. b and d.

## 16 C++ Question:

```
void foo(int i, double d, int j) {cout << "int, double, int" << endl; } // foo 1
void foo(double d, int i, int j) {cout << "double, int, int" << endl; } // foo 2
void foo(long l, int i, int j) {cout << "long, int, int" << endl; } // foo 3

int main( ) {
    long L = 0; int i = 0;
    float f; double d;
    foo(i, f, i); // call 1
    foo(f, L, 2); // call 2
    foo(1.0, 1.0, 2); // call 3
    foo(L, i, i); // call 4
}
```

Which answer is most correct?

- a. “call 1” calls “foo 1”, “call 2” calls “foo 2”, “call 3” is ambiguous, “call 4” calls “foo 3”.
- b. “call 3” is ambiguous, “call 2” is illegal because it passes a 64-bit “long” to a 32-bit “int”.
- c. “call 3” is ambiguous, “call 2” is illegal because it passes a 64-bit “long” to a 32-bit “int”.
- d. Only with exact matches (i.e. “call 4”) can C++ figure out which overloaded function to call.

## 17 Java Question:

```
class Test {  
  
    static void foo(int i, double d, int j) {System.out.println("int, double, int"); } // foo1  
  
    static void foo(double d, int i, int j) {System.out.println("double, int, int");} // foo 2  
  
    static void foo(long l, int i, int j) {System.out.println("long, int, int"); } // foo 3  
  
    public static void main(String args[]) {  
        long L = 0; int i = 0;  
        float f = (float) 0.0; double d = 0.0;  
        foo(i, f, i); // call 1  
        foo(f, L, 2); // call 2  
        foo(L, i, i); // call 3  
    }  
}
```

Which answer is most correct?

- a. “call 1” has no match, “call 2” calls “foo 2”, and “call 3” call “foo 2”.
- b. “call 1” calls “foo 1”, , “call 2” has no match, “call 3” calls “foo 3”
- c. “call 1” calls “foo 1”, , “call 2” calls foo2 and “call 3” is ambiguous.
- d. In Java, only calls whose arguments types exactly match the function parameter types are legal, and thus only “call 3” is legal.

## 18 Java Question:

```
class B {
    public B( ) {System.out.print("B ");}
}

class D1 extends B {
    public D1( ) {System.out.print("D1 ");}
}

class D2 extends D1 {
    public D2( ) {System.out.print("D2 ");}
}

class Main {
    public static void main(String args[]) {
        D2 d = new D2( );
    }
}
```

Which answer is most correct?

- a. The program prints “D2”
- b. When a constructor other than the default zero-arg constructor is used it must be called explicitly, and the program is illegal.
- c. The program prints “B D1 D2”
- d. The program prints “D2 D1 B”
- e. The order that constructors are called is implementation dependent, and either a, c or d could be correct.

## 19 C++ Question:

```
class B {
public:
    B( ) {cout << "B " ;}
};

class D1 : B {
public:
    D1( ) {cout << "D1 " ;}
};

class D2 : D1 {
public:
    D2( ) {cout << "D2 " ;}
};

int main( ) {
    D2* d = new D2( );
}
```

Which answer is most correct?

- a. The program prints “D2”
- b. When a constructor other than the default zero-arg constructor is used it must be called explicitly, and the program is illegal.
- c. The program prints “D2 D1 B”
- d. The program prints “B D1 D2”
- e. The order that constructors are called is implementation dependent, and either a, c or d could be legal.

## 20 C++ Question:

```
class B {
    int i;
    float f;
    double d;
public:
    B( ) : d(0.0), i(4), f(3.0) { }
};

int main( ) {
    B b( );
}
```

Which answer is most correct?

- a. When the zero-arg constructor for “B” is called, the order the fields are initialized is implementation dependent, but the same for all executions of the program.
- b. When the zero-arg constructor for “B” is called, the order the fields are initialized can vary from program run to program run.
- c. When the zero-arg constructor for “B” is called, field “d” is initialized first, then field “i” next, and then field “f”.
- d. When the zero-arg constructor for “B” is called, field “f” is initialized first, then field “i” next, and then field “d”.
- e. When the zero-arg constructor for “B” is called, field “i” is initialized first, then field “f” next, and then field “d”.

## 21 Java (+4 if you get it, -2 if you are wrong)

```
class B {
    private int i;
    public B( ) {i = 4;}
    void add(B b) {i = i + b.i;}
}

class Test {

    public static void main(String args[]) {
        B b1 = new B( );
        B b2 = new B( );
        b2.add(b1);
    }
}
```

Which answer is most correct?

- a. Because “i” is a private field of the B class, object “b2” can access object “b1”’s “i” field in the “add” method.
- b. Because “i” is a private field of the B object, object “b2” cannot access object “b1”’s “i” field in the “add” method.