ECE 30862 Fall 20l2, Third 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 17 pages total (including this cover sheet). *Please let Prof. Midkiff know immediately if it does not.* Each problem is worth 6.5 points.

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.*

All questions are worth 4 points.

Name:

1 C++ Question:

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

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

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

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

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

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

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

   D d = D( ); // create a object d of type D
   D* dp = (D*) &d;
   B* bp = (B*) dp;
   B& br = (B&) d; // br is a reference to the object pointed to by d.

   bp->print( );
   bp->whoAmI(1);
   (B (*dp)).print( );
   (B (*dp)).whoAmI(2);
   return 0;
}
```

What is printed?

a. B::print B::whoAmI B::print B::whoAmI

b. D::print B::whoAmI B::print B::whoAmI

c. D::print D::whoAmI D::print D::whoAmI

d. B::print B::whoAmI D::print B::whoAmI

e. Illegal program because the compiler cannot figure out whether to call B's print and whoAmI or D's.

2 C++ Question:

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

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

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

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

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

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

   vector<B*> vec;
   vec.push_back(new B( )); vec.push_back(new B( ));
   vec.push_back((B*) new D( )); vec.push_back((B*) new D( ));
   vec[0]->print( );
   vec[2]->print( );
   return 0;
}
```

In the question below, when I say that an element "points to an `X` object" I do not mean that the element is of type "`X*`", but that the object pointed to by the pointer held in the element is of type `X`.

a: vec[0] points to a B object, vec[2] points to a B object

b: vec[0] points to a D object, vec[2] points to a B object

c: vec[0] points to a D object, vec[2] points to a D object

d: vec[0] points to a B object, vec[2] points to a D object

e: The program is illegal because the vector can only hold pointers that point to B objects.

3 C++ Question:

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

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

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

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

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

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

   vector<B*> vec;
   vec.push_back(new B( )); vec.push_back(new B( ));
   vec.push_back((B*) new D( )); vec.push_back((B*) new D( ));
   vec[0]->print( );
   vec[2]->print( );
   return 0;
}
```

What is printed?

a: B::print B::print

b: D::print B::print

c: D::print D::print

d: B::print D::print

4 C++ Question:

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

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

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

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

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

   D* d = new D( );
   d->print( );

   return 0;
}
```

What is printed?

a. B::print

b. Nothing – there is an error because there is no print function

c. Because print is virtual, B::print is called.

d. a and c.

5 C++ Question:

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

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

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

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

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

   int i = 4;
   D* d = new D( );
   d->print(i);

   return 0;
}
```

What is printed?

a. B::print

b. D::print

6 C++ Question:

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

class B AAAA {
public:
   int age;
    B( ) {age = 20;}
};

class B1 : BBBB public B {
public:
   B1( ) { age=1; }
   ~B1( ) { };
};

class B2 : CCCC public B {
public:
   B2( ) { age=2; }
   ~B2( ) { };
};

class D : DDDD public B1, public B2 {
public:
   D( ) : B( ), B1( ), B2( )  {age=2; }
   ~D();
};

int main(int argc, char * argv[]) {
   D* d = new D( );
   cout << d->age << endl;
}
```

What should be substituted for "AAAA", "BBBB", "CCCC" and "DDDD" in the above program?

a. They should be blank as the `public` keywords have already be specified for inherited functions.

b. "virtual" should be substituted for all of "AAAA", "BBBB", "CCCC" and "DDDD" to keep multiple copies of `B` from being created in both `B1` and `B2`

c. "virtual" should be substituted for "BBBB" and "CCCC" to keep multiple copies of `B` from being created in both `B1` and `B2`

d. "virtual" should be substituted for "AAAA" to keep multiple copies of `B` from being created in both `B1` and `B2`

e. "virtual" should be substituted for "DDDD" to allow it to inherit only one copy of `B` from `B1` and `B2`

7 C++ Question:

This is the same program as on the previous question.

```
class B AAAA {
public:
   int age;
    B( ) {age = 20;}
};

class B1 : BBBB public B {
public:
   B1( ) { age=1; }
   ~B1( ) { };
};

class B2 : CCCC public B {
public:
   B2( ) { age=2; }
   ~B2( ) { };
};

class D : DDDD public B1, public B2 {
public:
   D( ) : B( ), B1( ), B2( )  {age=2; } // MENTIONED IN QUESTION
   ~D();
};

int main(int argc, char * argv[]) {
   D* d = new D( );
   cout << d->age << endl;
}
```

Pick the most correct answer about the statement with the comment "MENTIONED IN QUESTION".

a. The constructors for all directly and indirectly inherited functions are mentioned to make it clear to other programmers what classes are used by `D`.

b. The constructors mentioned in the initializer list are invoked in the order listed.

c. The constructors mentioned in the initializer list are invoked in the order the classes they are constructors for are declared.

d. The call to the constructor for `B( )` is necessary if it is virtually inherited and `D` is the most derived function in the chain inheriting from `B`.

e. b and d.

f. c and d.

8 Java Question:

```
class E0 extends Exception {

   public E0( ) { }
   public void print( ) {System.out.print("E0 ");}
}

class E1 extends Exception {

   public E1( ) { }
   public void print( ) {System.out.print("E1 ");}
}

class Test {

   private static void f(int i) throws E0, E1 {
      if (i == 0) throw new E0( );
      if (i == 1) throw new E1( );
   }

   public static void main(String args[]) {

      for (int i = 0; i < 3; i++) {
         try {
            f(i);
         } catch (E0 e) {
            e.print( );
         } catch (E1 e) {
            e.print( );
         }
      }
      System.out.println("terminating program");
   }
}
```

What is the best answer?

a. This is an illegal program because there is no "finally" clause.

b. The program prints "E0 ".

c. The program prints "E0 E1 ".

d. The program prints "E0 terminating program".

e. The program prints "E0 E1 terminating program".

9 Java Question:

```
class E0 extends Exception {

   public E0( ) { }
   public void print( ) {System.out.print("E0 ");}
}

class E1 extends Exception {

   public E1( ) { }
   public void print( ) {System.out.print("E1 ");}
}

class Test {

   private static void f(int i) throws E0, E1 {
      if (i == 0) throw new E0( );
      if (i == 1) throw new E1( );
   }

   public static void main(String args[]) {

      for (int i = 0; i < 3; i++) {
         try {
            f(i);
         } catch (E0 e) {
            e.print( );
         } catch (E1 e) {
            e.print( );
         } finally {System.out.print("F ");}
      }
      System.out.println("The end");
   }
}
```

What is the best answer?

a. "E0 F" is printed.

b. "E0 F The end" is printed.

c. "E0 F E1 F F The end" is printed.

d. "E0 F E1 F " is printed.

e. "E0 F E1 F The End " is printed.

10 C++ Question:

```
class E {
public:
    E( ) { }
   ~E( ) { }
   void print( ) {cout << "E ";}
};

void f(int i) {if (i < 0) {throw E( );} }

int main( ) {
   for (int i = -1; i < 2; i++) {
      try {
         f(i);
         cout << i << " ";
      } catch (E e) {
         e.print( );
      }
   }
   cout << "the end" << endl;
}
```

Which answer is most correct?

a. This is an illegal program because C++ only allows primitives (such as integers) to be thrown.

b. "E " is printed.

c. "E 0" is printed.

d. "E 0 the end" is printed.

e. "E 0 1 the end" is printed.

11 Threads Question:

Assume that variables X and Y are initialized to 0 when the program begins. Consider the following code executing in *Thread 1* and *Thread 2*. There is no synchronization in the program but you may assume all statements in the thread execute in the order written, i.e. the execution is sequentially consistent.

Thread 1	Thread 2
`X = 1;`	`X = 2;`
`Y = 1;`	`Y = 2;`

Which answer is most correct about what the values of `X` and `Y` can be after the code in Thread 1 and Thread 2 executes?

a. "X=1, Y=1"

b. "X=2,Y=2"

c. "X=2, Y=1"

d. "X=1, Y=2"

e. Any of a, b, c or d.

12 Threads Question:

Assume that variables X and Y are initialized to 0 when the program begins. Consider the following code executing in *Thread 1* and *Thread 2*.

Thread 1	Thread 2
`sychronized(L1) {`	`sychronized(L1) {`
`  X = 1;`	`  X = 2;`
`  Y = 1;`	`  Y = 2;`
`}`	`}`

Which answer is most correct about what the values of `X` and `Y` can be after the code in Thread 1 and Thread 2 executes??

a. "X=1, Y=1"

b. "X=2,Y=2"

c. "X=2, Y=1"

d. "X=1, Y=2"

e. Either a or b.

f. Any of a, b, c or d.

13 Java Question:

```
class B {
   void print(float f) {System.out.println("B::print(float)");}
}

class D extends B {
   void print(int i) {System.out.println("D::print(int)");}
   void print(long l) {System.out.println("D::print(long)");}
}

class Test {

   public static void main(String args[]) {
      int i = 0;
      float f = (float) 1.0;
      D d = new D( );
      B b = (B) d;
      d.print(i);
      b.print(i);
   }
}
```

What is printed or is otherwise the best answer?

a. **D::print(int) B::print(float)**

b. D::print(int) B::print(int)

c. D::print(int) D::print(int)

d. D::print(int) D::print(float)

e. Illegal program because you cannot override overloaded methods.

14 Java Question:

```
class B {
   void print(float f) {System.out.println("B::print(float)");}
}

class D extends B {
   void print(int i) {System.out.println("D::print(int)");}
   void print(long l) {System.out.println("D::print(long)");}
   void print(float f) {System.out.println("D::print(float)");}
}

class Test {

   public static void main(String args[]) {
      int i = 0;
      float f = (float) 1.0;
      D d = new D( );
      B b = (B) d;
      d.print(i);
      b.print(i);
   }
}
```

What is printed or is otherwise the best answer?

a. D::print(int) B::print(int)

b. **D::print(int) D::print(int)**

c. D::print(int) B::print(float)

d. D::print(int) D::print(float) 

e. Illegal program because you cannot override overloaded methods.

15 C++ Question:

```
class Quadrilateral {
public:
   virtual double area(int w, int h) {return w*h;}
   virtual double perimeter() = 0;
};

class Square : Quadrilateral {
public:
   . . .
}

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

Which answer is most correct?

a. Class `Quadrilateral` is abstract, only `perimeter` is an abstract function, and therefore `Square` must only define `perimeter`.

b. Class `Quadrilateral` is abstract, and class `Square` must define the area and `perimeter` functions.

c. `perimeter` will always returned zero if not defined in class `Square`.

d. Class `Quadrilateral` is abstract, an `perimeter` needs to be defined only if it will be used by class `Square`.

16 C++ Question:

```
class MyComplex {
public:
   double re, im;
   MyComplex(double r, double i) : re(r), im(i) { }
   MyComplex operator-(arguments); //
};

int main( ) {
   . . .
}
```

Which answer is most correct?

a. The parameter list for function `MyComplex::operator-` must be "`const MyComplex& arg`" because the function is a member of the MyComplex class.

b. The parameter list for function `MyComplex::operator-` must be "`const MyComplex& arg, const MyComplex& arg`" because "-" is a binary operator.

c. The parameter list for function `MyComplex::operator-` can be empty if the function is a unary "-" operation and "`const MyComplex& arg, const MyComplex& arg`" if the function is for a binary operator.

d. The parameter list for function `MyComplex::operator-` can be "`const MyComplex& arg, const MyComplex& arg`" if the function is for a unary operator and "`const MyComplex& arg, const MyComplex& arg`" if it is a binary operator.