ECE 461 Fall 2011, Final Exam

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

You have until 9:00PM 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.*

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:

Q1 (4.5 points):
Circle the most correct answer:

a. A `friend` function can access all protected fields and functions in the class that declares it as friend.

b. A `friend` function can access all private fields and functions in the class that declares it as friend.

c. A `friend` function can access all public fields and functions in the class that declares it as friend.

d. A `friend` function can access all public, protected and private fields and functions in the class that declares it as friend.

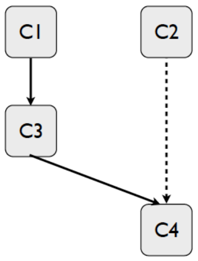
e. A `friend` function declared in some class `C` can only accesss fields in classes that `C` inherits from, directly or indirectly.

e. A `friend` function declared in some class `C` can only accesss fields in classes that inherit from `C`, directly or indirectly.

Q2 (4.5 points):
Fill in the underlined areas in the Java program outlined below to reflect the inheritance shown in the drawing to the right. Leave it blank if nothing else is needed. I use a solid line to indicate inheritance of a class, and a dashed line to indicate inheritance or implementation of an abstract class or interface.

A Java program outline:
```
class C1 ________________ {
...
}
class C2 ________________ {
...
}
class C3 extends C1 {
...
}
class C4 extends C3, implements C2 {
...
}
```

A C++ program outline:
```
class C1 ________________ {
...
}
class C2 ________________ {
...
}
class C3 : C1 {
...
}
class C4 : C3, C2 {
...
}
```



Q3 (4.5 points):
Consider the UML diagram below. The class `C1` is multiply inherited by class `C4`. Circle the most correct answer below.

a. In C++, `C2` and `C3` must virtually inherit from `C1` to keep two copies of `C1` from being created.

b. In Java, `C1` must be declared `final` so that it is only created once.

c. In Java this is not a problem because multiple inheritance is not allowed.

d. Java does not allow virtual inheritance, and the programmer must explicitly specify which copy of `C1` is being accessed whenever a field or function declared in `C1` is accessed.

[✓ **e.** Both (a) and (c).]

$$\begin{tikzcd} & C1 \arrow[dl] \arrow[dr] & \\ C3 \arrow[dr] & & C2 \arrow[dl] \\ & C4 & \end{tikzcd}$$

Q4 (4.5 points):

```
class B {
   int i = 0;
   public B(int i) {this.i = i;}
   public void incr(int j) {i += j;}
   public int getI( ) {return i;}
   public void setI(int j) {i = j;}
}

class D extends B {
   public D(int i) {super(i);}
   public void incr(int j) {
      int m = super.getI( ) + j;
      super.setI(m);
   }
   public void incr(int j, int k) {
      int m = j + k;
      super.setI(m);
   }
   public int getVal( ) {return super.getI( );}
}

class Main {
   public static void main(String [] str) {
      D d = new D(5);
      d.incr(5);
      System.out.println("val 1 "+d.getVal( ));
      d.incr(5,6);
      System.out.println("val 2 "+d.getVal( ));
   }
}
```

The outcome of this program is:

a. A compile time error because the call `d.incr(5);` is ambigious because a function "`public void incr(int)`" is declared in both class `B` and class `D`

b. A compile time error because there are two functions named `incr` in class `D`

[✓ **c.** `val 1 10` and `val 2 11` are printed.]

d. A runtime error (i.e. error during execution) because there are two functions named `incr` in class `D`

e. `val 1 10` and `val 2 21` are printed.

f. An error because there is no function named `super` declared in class `B` or `D`

Q5 (4.5 points):
In a C++ program class D inherits from class B. Both B and D implement a member function (i.e. the function is not static) `foo(int)`. The code, including D's `foo` function, looks like:

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

class B {
private:
   int i;
public:
   virtual int bar(int k) {
      return 2*k;
   }
};

class D : B {
int j;
public:
   virtual int foo(int j) {
      // Need to call B's ~~foo~~ and pass the argument j to B's ~~foo~~      j= bar(j);
      // Assign the return value from B's ~~foo to~~ the D object's j
   }
};
```

What should replace the commented text to implement the needed functionality?

Q6 (4.5 points):

Make the following Java class abstract.

```
abstract class Abs {
   int j = 49;
   int k = 0;
}
```

Q7 (4.5 points):
A programmer reads in a list of integers, some of which might be duplicates. The programmer then wants to print the strings without duplicates. What is a good container to use for this and why?

A set -- no duplicates, O(1) inserts

Q8 (4.5 points):
Consider the following program:

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

class X {
public:
   X( ) {
      cout << " X ";
   }
};

class Y : public X {
public:
   Y( ) {
      cout << " Y ";
   }
};

int main( ) {
   Y* y = new Y( );
   cout << endl;
}
```

What is printed?

a. X

b. Y

[✓ **c.** X Y]

d. Y X

Q9 (4.5 points):
Consider the following program:

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

class X {
int j;
public:
   X(int i) {
      j = i;
      cout << " X " << endl;
   }
};

class Y : public X {
public:
   Y( ) {
      cout << " Y ";
   }
};

int main( ) {
   Y* y = new Y( );
   cout << endl;
}
```

What is printed?

a. `X`

b. `Y`

c. `X Y`

d. `Y X`

e. The program does not compile because no zero argument constructor is supplied for class `X`.

Q10 (4.5 points):
Programmer Mary is asked to make some changes to the program of Q9, and after deleting a few lines comes up with the following program:

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

class X {
int j;
public:
};

class Y : public X {
public:
   Y( ) {
      cout << " Y ";
   }
};

int main( ) {
   Y* y = new Y( );
   cout << endl;
}
```

When she compiles it, and, if it compiles correctly runs it, is the outcome?

a. It prints `Y`

b. The program does not compile because no zero argument constructor is s upplied for class `X`.

c. The program segmentation faults after evaluating the left shift `cout << endl;`.

Q11 (4.5 points):

Show the code requested in the comment in `foo`.

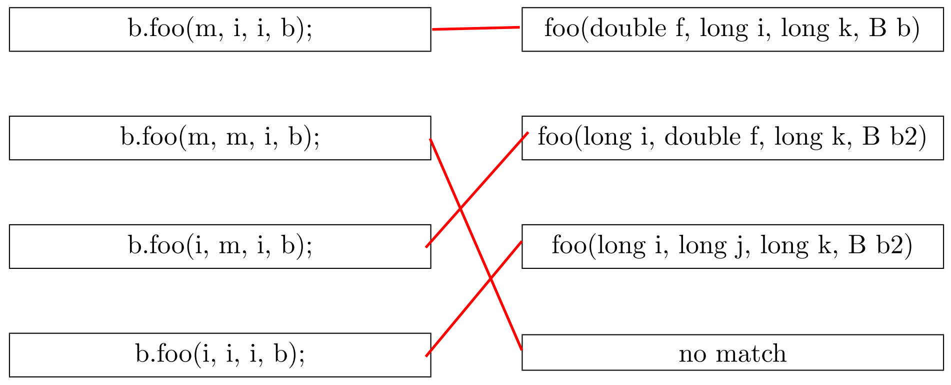
```
#include <string>
using namespace std;

class X {
int i;
public:
   int foo(int j) {
      return 2*j;
   }
   // add a declaration here that allows global function bar to access i.
   // if no declaration is needed, say "not needed".        friend int bar( X x );
};

int bar(X b) {
   b.i = 5011;
   return b.i;
}
```

Consider the following Java program is used for the question below.

```
class B {

   public B( ) { }

   public void foo(double f, long i, long k, B b) {
      System.out.println("foo int double B2");
   }

   public void foo(long i, double f, long k, B b2) {
      System.out.println("foo int int B2");
   }

   public void foo(long i, long j, long k, B b2) {
      System.out.println("foo double int B2");
   }
}

public class Main {

   public static void main(String args[ ]) {
      int i=0;
      long j=1;
      double m=1.1;

      B b = new B( );
      b.foo(m, i, i, b);
      b.foo(m, m, i, b);
      b.foo(i, m, i, b);
      b.foo(i, i, i, b);
   }
}
```

Q12 (4.5 points):

Match the calls on the left with the function header of the invoked function on the right. If there is no matching function, match it with "none matched."


b.foo(m, i, i, b);
foo(double f, long i, long k, B b)
b.foo(m, m, i, b);
foo(long i, double f, long k, B b2)
b.foo(i, m, i, b);
foo(long i, long j, long k, B b2)
b.foo(i, i, i, b);
no match

Q13 (4.5 points):
The following Java program is identical to the one in Q12, and is reproduced here for your convenience.

```
class B {

   public B( ) { }

   public void foo(double f, long i, long k, B b) {
      System.out.println("foo int double B2");
   }

   public void foo(long i, double f, long k, B b2) {
      System.out.println("foo int int B2");
   }

   public void foo(long i, long j, long k, B b2) {
      System.out.println("foo double int B2");
   }
}

public class Main {

   public static void main(String args[ ]) {
      int i=0;
      long j=1;
      double m=1.1;

      B b = new B( );
      b.foo(m, i, i, b);
      b.foo(m, m, i, b);
      b.foo(i, m, i, b);
      b.foo(i, i, i, b);
   }
}
```

The compiler complains about not finding a matching symbol for at least one of the calls. Show a call that it complains about and put in casts for the call to make it legal.

Either
b.foo(m, (long) m, i, b); or
b.foo((long) m, m, i, b);

Q14 (4.5 points):

What is printed by the program below?

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

using namespace std;

class C {
public:
   int f;
   C( ){ };
   C(int i) {f=i;};
   ~C( ){ };
};

class B {
public:
   B( ){ };
   ~B( ){ };
   void foo(C& c, int i) {
      c.f = -10;
      i = c.f;
   }
};

int main(int argc, char * argv[ ])
{
   int i = 42;
   B* b = new B( );
   C c1(2);
   C& c = (C&) c1;
   b->foo(c, i);
   cout << c.f << " " << i << endl;
}
```

-10 42

Q15 (4.5 points):

What is printed by the program below?

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

class MyComplex {
private:
   double re, im;
public:
   MyComplex(double r, double i) : re(r), im(i) { };
   MyComplex operator-(MyComplex) const;
   friend ostream& operator<< (ostream&, const MyComplex&);
};

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

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

int main( ) {
   MyComplex u(3,4);
   MyComplex v(2,9);

   cout << u + v << endl;
   return 0;
}
```

1 13

Q16 (6 points):

Why does " <<" need to be declared as a `friend` function in the program for Q15?

so that it can access the private re and im fields.

Q17 (4.5 points): What is printed by the following program:

```
import java.io.*;

class Main {
static void f(int j) throws Exception {
   System.out.println( j );
   throw new Exception();
}

public static void main( String[] args ) {
   try {
      f(0);
   } catch(Exception e ) {
      System.out.println("Exception caught");
   } finally {
      System.out.println("in Finally");
   }
}

}
```

0
Exception caught
in Finally

Q18 (4.5 points):

What is printed by the following program:

```
class Except extends Throwable {
   public Except( ) { }
   public void printError( ) {
      System.out.println("Except");
   }
}

class Bad1 extends Except {
   public Bad1( ) { }

   public void printError( ) {
      System.out.println("Bad1");
   }
}

class Bad2 extends Except {
   public Bad2( ) { }

   public void printError( ) {
      System.out.println("Bad2");
   }
}

class Main {

   static void f(int j) throws Except {
      System.out.println( j );
      if (j == 1) throw new Bad1();
      if (j == 2) throw new Bad2();
      throw new Except();
   }

   public static void main( String[] args ) {
      try {
         f(1);
      } catch(Except e ) {
         e.printError( );
      } finally {
         System.out.println("in Finally");
      }
   }
}
```

1
Except
in Finally

Q19 (4.5 points):
Is there a race on the accesses to the objects `i0` and `i1` in the `run` methods of `t0` and `t1` in the program below? Why or why not??

```
class I {
   public int i;
   I( ) {i = 0;}
   I(int a) {i = a;}
}

class T extends Thread {
   I p1, p2;

   T() { };
   T(I a1, I a2) {
       p1 = a1; p2 = a1;
   };

   public void run( ) {
      synchronized(p1) {
         synchronized(p2) {
            int tmp = p1.i;
            p1.i = p2.i;
            p2.i = tmp;
         }
      }
   }
}

class Main {

   public static void main( String[] args ) {
      I i0 = new I(0);
      I i1 = new I(1);
      T t0 = new T(i0,i1);
      T t1 = new T(i1,i0);
      t0.start( );
      t1.start( );
      System.out.println("all done"+": "+i0.i+", "+i1.i);
   }
}
```

Yes -- referenes to the same objects (i0 and i1) are passed to two threads which both read and write the objects.

Q20 (4.5 points):
Can the program below deadlock? (This program is identical to the one found in Q19 and is reproduced here for your convenience.) If it can, fill in the table below with the sequence of synchronization statements that will cause it to deadlock. The first entry is filled in as an example of how you should write the statements. If it cannot deadlock, say why deadlock is impossible.

step 1	t0.synchronized(p1) [i0 object]
step 2	t1.synchronized(p1) [i1 object]
step 3	t0 tries to acquire p2 [i1 object]
step 4	t1 tries to acquire p2 [i0 object]

```
class I {
   public int i;
   I( ) {i = 0;}
   I(int a) {i = a;}
}
class T extends Thread {
   I p1, p2;

   T() { };
   T(I a1, I a2) {
       p1 = a1; p2 = a1;
   };

   public void run( ) {
      synchronized(p1) {
         synchronized(p2) {
            int tmp = p1.i;
            p1.i = p2.i;
            p2.i = tmp;
         }
      }
   }
}
class Main {

   public static void main( String[] args ) {
      I i0 = new I(0);
      I i1 = new I(1);
      T t0 = new T(i0,i1);
      T t1 = new T(i1,i0);
      t0.start( );
      t1.start( );
      System.out.println("all done"+": "+i0.i+", "+i1.i);
   }
}
```

The next two questions use the program below.

```
#include <string>
#include <iostream>using namespace std;
class B {
public: int i;
   B( ) {};

class D1 : _________________ B {
public:
   int j;
   D1( ) : B( ) {j=1;}
};

class D2 : public D1 {
public:
   D2( ) : D1( ) { }
   void foo( ) {i = 20;}
};

int main(int argc, char * argv[ ]) {
   D1* d1 = new D2( );
   d1->i; \\ *** S1 ***
   D2* d2 = new D2( );
   . . .
}
```

Q21 (4.5 points):
The designer of class `D1` wants classes that inherit from it to be able to access the `i` variable in class `B`, but *NOT* allow uses of a `D1` object (as in statement `S1` in function `main`) to access `i`. What keyword should be put in the line in the header of `D1`? protected

Q22 (4.5 points):
The designer of class `D1` doesn't want classes that inherit from it to access the `i` variable in class `B`, and doesn't want to allow accesses like those in statement `S1` of `main`. What keyword should be put in the line in the header of `D1` to enforce these restrictions? private