ECE 30862 Fall 2014, Final Exam

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

THE LAST TWO PAGES ARE THE ANSWER SHEETS. TEAR THEM OFF AND PUT ALL ANSWERS ON THEM. TURN IN BOTH ANSWER SHEETS AND THE TEST ITSELF WHEN FINISHED.

You have until 10:00AM to take this exam. Each question is work 1.3 points unless otherwise noted. The total number of points should be 100. After taking the test turn in both the test and the answer sheet.

Your exam should have 12 (twelve) pages total (including this cover page). As soon as the test begins, check that your exam is complete and *let the proctors know immediately if it is not.*

This exam is open book, open notes, but absolutely 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.*

I have neither given nor received help during this exam from any other person or electronic source, and I understand that if I have I will be guilty of cheating and will fail the exam and perhaps the course.

Name (must be signed to be graded):

Name

Last four digits of your ID:

Java function call questions. 1

```
class B {

   public char c = 'B';

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

   private void priv( ) {
      System.out.println("B");
   }

   public void printPriv(B b) {
      b.priv( );
   }

   public void callPrint( ) {
      print((int) 4, (long) 5);
   }

   public void print(int i, long ii) {
      System.out.println("bil");
   }

   public void print(long ii, long jj) {
      System.out.println("bll");
   }
}
```

```
class D extends B {

   public char c = 'D';

   public D( ) {System.out.print(" D ");}

   private void priv( ) {
      System.out.println("D");
   }

   public void printPriv(D d) {
      d.priv( );
   }

   public void print(int i, long ii) { // LINE A
      System.out.println("dil");
   }

   public int print(int i, long ii) { // LINE B
      System.out.println("dil");
   }
}
```

```
class Key {

   public static void main(String args[]) {

      B b = new B( );
      D d = new D( ); // Q2

      d.print((long) 4, (long) 5); // Q3
      d.print((int) 4, (int) 5); // Q4
      d.print((int) 4, (long) 5); // Q5
      d.printPriv(d); // Q6
      d.printPriv(b); // Q7

      b.print((long) 4, (long) 5); // Q8
      b.print((int) 4, (int) 5); // Q9
      b.print((int) 4, (long) 5); // Q10
      b.printPriv(d); // Q11
      b.printPriv(b); // Q12

      System.out.println(d.c); // Q13
      System.out.println(b.c); // Q14

      b = d;
      System.out.println(b.c); // Q15

      b.print((int) 4, (int) 5); // Q16
      b.print((int) 4, (long) 5); // Q17
      b.printPriv(d); // Q18
      b.printPriv(b); // Q19

      b.callPrint( ); // Q20
   }
}
```

Q1 Is LINE B a legal declaration of a function? Pick the answer below that is most correct.

a. Yes, because even though the arguments are the same as in LINE A (the function declaration immediately before this one) the return types are different.

b. No, because the function name and the arguments are the same as in LINE A.

c. Yes, as long as the function is not called with a cast to a return type other than `void` or `int`, as this would make it ambiguous which one to call.

Q2 – Q20 Say what is printed by each line followed by a comment containing a question number. If the line is an error, answer "E". If the line does not print anything and is not an error enter "OK".

Java exceptions questions. [2]

```
class EB extends Exception {

   String str;

   public EB( ) {
      str = "B";
   }
}

class ED extends EB {

   String str;

   public ED( ) {
      str = "D";
   }
}
```

```
class Test {

   public static void thrower(int i) throws EB, ED {
      if (i == 0) throw new EB( );
      else if (i == 1) throw new ED( );
   }
}

class Main {

   public static void main(String args[]) {

      int i = 0;
      while (i < 2) {
         try { // A
            Test.thrower(i);
         } catch (ED e) {System.out.print(e.str); // LINE A
         } catch (EB e) {System.out.print(e.str); // LINE B
         } finally {System.out.print(" final "); i++;}
      }
      System.out.println(" ");
   }
}
```

21. What is printed by the program?

a. B final D final

b. D final B final

c. D final

d. B final

e. B D final

f. D B final

22. If "LINE A" and "LINE B" are reversed in the program, i.e. LINE B is first followed by LINE A, what will be the result?

a. No problem because exceptions are caught precisely by type of the exception

b. There will an error because the base class exception **EB** will always be caught since its catch clause (in LINE B) is now first and the code in LINE A is never reached.

c. The program will be undefined.

Java multithreading questions. 3

```
class MyThread extends Thread {

   public static int count = 0;
   // public static Object lock = new Object( ); // LINE A

   public MyThread( ) { };

   public void run( ) {
      for (int i = 0; i < 1000; i++) {
      //   synchronized(lock) { // LINE B
            int j = count;
            try {
               // thread sleeps for approx. 1 millisecond
               Thread.sleep(1);
            } catch (java.lang.InterruptedException e) { }
            count = j+1;
       //  } // LINE C
      }
   }
}
```

```
class Main {

   public static void main(String args[]) {

      MyThread[ ] threads = new MyThread[4];
      for (int i = 0; i < 3; i++) {
         threads[i] = new MyThread( );
      }

      for (int i = 0; i < 3; i++) {
         threads[i].start( ); // LINE D
         threads[i].run( );
      }

      for (int i = 0; i < 3; i++) {
         try {
            threads[i].join( );
         } catch (java.lang.InterruptedException e) {
            System.out.println(e);
         }
      }

      System.out.println(MyThread.count);
   }
}
```

23. For the program above, which statement is most true?

a. The final value of `Mythread.count` can be less than 6000.

b. The final value of `Mythread.count` can be greater than 6000.

c. The final value of `Mythread.count` will always be equal to 6000.

24. In the program above, consider the program when "LINE A", "LINE B" and "LINE C" are uncommented. Now, which statement below is most true?

a. The final value of `Mythread.count` can be less than 6000.

b. The final value of `Mythread.count` can be greater than 6000.

c. The final value of `Mythread.count` will always be equal to 6000.

25. In the program above, assume "LINE A", "LINE B" and "LINE C" are commented out. if "LINE D" is changed to `threads[i].run( );`, i.e., `start` is changed to `run`, what statement is most true?

a. The final value of `Mythread.count` can be less than 6000.

b. The final value of `Mythread.count` can be greater than 6000.

c. The final value of `Mythread.count` will always be equal to 6000.

C++ function call questions. [4]

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

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

   virtual void callPrint(int i) {print(i);}

   virtual void foo(float i) {
      cout << "Bi" << endl;
      cout << endl;
   }

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

};

class D : public B {
public:
   D( ) { }
   virtual ~D() { }
   virtual void callPrint(int i) {print(i);}

   virtual void foo(int i) {
      cout << "Di" << endl;
      cout << endl;
   }

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

   virtual void bar( ) {
      cout << "D" << endl;
   }
};
```

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

   B b = B( );
   D d = D( );
   B* bp = &b;
   D* dp = &d;
   B& br = b;
   D& dr = d;

   b.bar( ); // Q26

   d.callPrint(1); // Q27
   d.foo(1.0); // Q28
   d.print(1); // Q29

   dp->callPrint(1); // Q30
   dp->foo(1.0); // Q31

   b = d;
   bp = dp;
   br = dr;

   b.callPrint(1); // Q32
   b.foo(1); // Q33

   bp->callPrint(1); // Q34
   bp->foo(1); // Q35
   bp->print(1); // Q36

   br.callPrint(1); // Q37
   br.print(1); // Q38
   br.bar( ); // Q39

   return 0;
}
```

Q26 – Q39. Say what is printed by each line followed by a comment containing a question number. If the line is an error, answer "E". If the line does not print anything and is not an error enter "OK".

C++ function call questions. 5

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

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

   void callPrint(int i) {print(i);}

   static void foo(float i) {
      cout << "Bi" << endl;
      cout << endl;
   }

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

};

class D : public B {
public:
   D( ) { }
   virtual ~D() { }
   virtual void callPrint(int i) {print(i);}

   static void foo(int i) {
      cout << "Di" << endl;
      cout << endl;
   }

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

};
```

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

   B b = B( );
   D d = D( );
   B* bp = &b;
   D* dp = &d;

   d.callPrint(1); // Q40
   d.foo(1.0); // Q41

   b = d;
   bp = dp;

   b.callPrint(1); // Q42

   bp->callPrint(1); // Q43
   bp->print(1); // Q44
   return 0;
}
```

Q40 – Q44 Say what is printed by each line followed by a comment containing a question number. If the line is an error, answer "E". If the line does not print anything and is not an error enter "OK".

C++ Inheritance and privacy questions. [6]

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

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

   int i;

   virtual void callPrint(int i) { }

   virtual void foo(float f) {
      i = 2; // QX
      j = 2; // QX
      bar(j); // QX
   }

private:
   int j;

   virtual void bar(int i) { }

};

class D : private B {
public:
   B bPublic;
   D(B b) {bPublic = b;}
   virtual ~D() { }
   virtual void update( ) {
      bPublic.i = 2; // Q45
      bPublic.j = 3; // Q46
      bPublic.foo(2.0); // Q47
   }
};
```

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

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

   b.i = 2; // Q48
   b.j = 3; // Q49

   d.i = 2; // Q50
   d.j = 3; // Q51
   d.foo(2.0); // Q52

   return 0;
}
```

Q45 – Q52. For each line that is a question, say what is printed. If it gives an error answer "E". If it prints nothing but is legal answer "OK".

C++ Constructor and destructor questions. [7]

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

class C {
public:
   C( ) { };
   virtual ~C( ) {cout << " ~C ";}
};

class B {
public:
   C* c;
   B() {
      cout << " B ";
      c = new C( );
   }
   virtual ~B() {
      cout << " ~B ";
      delete c;
   }
};

class D : public B {
public:
   int i;
   int j;
   int k;

   D( ) : k(4), i(k), j(i) {
      cout << i << " " << j << " " << k; // Q53
   }

   D(int z) : k(z), i(z), j(z) {cout << " D "; }
};

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

   B* b = new B( ); // Q54
   cout << endl;
   D* di = new D(1); // Q55
   cout << endl;
   D dv = D(1);
   delete b; // Q56
   cout << endl;
   delete di; // Q57
   cout << endl;
} // Q58
```

Q53 – Q58. Say what is printed by each line followed by a comment containing a question number. If the line is an error, answer "E". If the line does not print anything and is not an error enter "OK". For For Q53, assume that any uninitialized variable has the value of 0 (zero). For Q58, give the results of any objects that were not previously deleted by are popped off the stack as a result of exiting `main`.

C++ Template questions s

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

template <typename T> class Wrapper {
public:
   Wrapper(const T& data);
   virtual ~Wrapper( );
   T& getData( );
   bool operator<(const Wrapper<T>&) const;
   bool operator==(const Wrapper<T>&) const;
   friend ostream& operator<< (ostream& os, Wrapper<T>& n) {
      os << n.data;
      return os;
   }

private:
   T data;
};

template <typename T> Wrapper<T>::Wrapper(const T& data) : data(data) { }

template <typename T> Wrapper<T>::~Wrapper( ) { }

template <typename T> T& Wrapper<T>::getData( ) {
   return data;
}

template <typename T> bool Wrapper<T>::operator<(const Wrapper<T>& n) const {
   return data < n.data;
}

template <typename T> bool Wrapper<T>::operator==(const Wrapper<T>& n) const {
   return data == n.data;
}
```

```
int main(void) {

   Wrapper<int> i = Wrapper<int>(4); // LINE A
   Wrapper<float> f = Wrapper<float>(5.1); // LINE B

   int z = i.getData( );
   float y = f.getData( );

   cout << i << endl;
   cout << f << endl;

   return 0;
}
```

For each question below, answer whether the function header is a result of "LINE A", "LINE B" or "Neither".

Q59. `bool operator<(const Wrapper<float>&) const;`

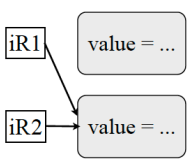
Q60. `bool operator==(const Wrapper<int>&) const;`

C++ parameter passing and reference variable questions. 9

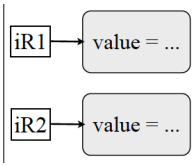
```
#include <iostream>
#include <string>
using namespace std;

class I {
public:
   int value;

   I(int v) {
      value = v;
   }

   I(const I& i) {
      cout << " I ";
      value = i.value;
   }

   virtual ~I() { }

   static void swapO(I i1, I i2) {
      int t = i1.value;
      i1.value = i2.value;
      i2.value = t;
      cout << i1.value << " " << i2.value << endl;
   }

   static void swapR(I& i1, I& i2) {
      int t = i1.value;
      i1.value = i2.value;
      i2.value = t;
      cout << i1.value << " " << i2.value << endl;
   }

   static void swapP(I* i1, I* i2) {
      int t = i1->value;
      i1->value = i2->value;
      i2->value = t;
      cout << i1->value << " " << i2->value << endl;
   }
};
```

```
int main(int argc, char * argv[]) {
   I* iP1 = new I(1);
   I* iP2 = new I(2);
   I iO1 = I(10);
   I iO2 = I(20);
   I& iR1 = *iP1;
   I& iR2 = *iP2;

   I::swapO(iO1, iO2); // Q61
   cout << iO1.value << " " << iO2.value << endl; // Q62

   I::swapR(iR1, iR2);
   cout << iR1.value << " " << iR2.value << endl; // Q63
   cout << iP1->value << " " << iP2->value << endl; // Q64

   I::swapP(iP1, iP2);
   cout << iR1.value << " " << iR2.value << endl; // Q65
   cout << iP1->value << " " << iP2->value << endl; // Q66

   iR1 = iR2; // LINE A

   return 0;
}
```

Q61 – Q66. Say what is printed by each question. Answer "E" if an error.



A



B

Q67. Refer to LINE A of the program and the drawings above when answering this question. Which figure best illustrates what `iR1` and `iR2` above reference after executing LINE A?

a. Option "A" in the drawings above.

b. Option "B" in the drawings above.

c. Neither of these.

C++ Operator overloading questions. [10]

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

class I {
public:
   int value;

   I(int v) {
      value = v;
   }

   I(const I& i) {
      cout << " I ";
      value = i.value;
   }

   virtual ~I() { }

   I& operator+(const I& i) const {
      I* tmp = new I(i.value + value);
      return *tmp;
   }

   I& operator-(const I& i) const {
      I* tmp = new I(value - i.value);
      cout << " - " << i.value << " " << value << endl;
      return *tmp;
   }

   friend ostream& operator<< (ostream& os, const I& i);

};

ostream& operator<< (ostream& os, const I& i) {
   os << i.value;
   return os;
}

I& operator-(I& i) {
   I* tmp = new I(-i.value);
   cout << " - " << i.value << endl;
   return *tmp;
}

int main(int argc, char * argv[]) {
   I i1 = I(1);
   I i2 = I(2);

   i2 = i1 + i2;
   cout << i1 << " " << i2 << endl; // Q68

   i2 = i2 - i1;
   cout << i1 << " " << i2 << endl; // Q69

   i2 = -i1; // Q70

   return 0;
}
```

Q68 – Q70. Say what is printed by each line followed by a comment containing a question number. If the line is an error, answer "E". If the line does not print anything and is not an error enter "OK".

Q71. In the function prototype `I& operator+(const I& i) const` the first use of `const` (in bold) says that

a. The object pointed to by the *this* pointer will not be changed.

b. Is a hint to the programmer writing the function not to change the value of the *this* pointer, but is not enforced.

c. The object referenced by `i` will not be changed.

d. Is a hint to the programmer writing the function not to change the value of the `i` argument, but is not enforced.

e. Says that the function should not be "changed" by a derived class and there will not be overridden.

Q72. In the function prototype `I& operator+(const I& i)` **const** the second use of `const` (in bold) says that

a. The object pointed to by the *this* pointer will not be changed.

b. Is a hint to the programmer writing the function not to change the value of the *this* pointer, but is not enforced.

c. The object referenced by `i` will not be changed.

d. Is a hint to the programmer writing the function not to change the value of the `i` argument, but is not enforced.

e. Says that the function should not be "changed" by a derived class and there will not be overridden.

C++ Exceptions questions. 11

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

class EB {
public:
   int v;
   EB( ) {v = -10;}
   EB(int i) : v(i) { }
};

class ED : public EB {
public:
   int v;
   ED(int i) : EB(-i), v(i) { }
};

class Thrower {

public:
   static void hurl(int i) {
      if (i == 2) throw EB(2);
      else if (i == 1) throw ED(1);
      else if (i == 100) throw 100; // LINE A
   }
};
```

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

   for (int i = 0; i < 3; i++) { // LOOP B
      try {
         Thrower::hurl(i);
      } catch (EB e) {cout << e.v << " ";
      } catch (ED e) {cout << e.v << " ";
      }
   }
   cout << endl;

   for (int i = 0; i < 3; i++) { // LOOP C
      try {
         Thrower::hurl(i);
      } catch (ED e) {cout << e.v << " ";
      } catch (EB e) {cout << e.v << " ";
      }
   }
   cout << endl;

   Thrower::hurl(100); // LINE D
   cout << "excption thrown" << endl; // LINE E

   return 0;
}
```

Q73. What is printed during the execution of LOOP B? If the loop gives an error, answer "E". If the loop does not print anything and is not an error enter "OK".

Q74. What is printed during the execution of LOOP C? If the loop gives an error, answer "E". If the loop does not print anything and is not an error enter "OK".

Q75. Pick the most correct answer:

a. "LINE A" gives a compile time error because only objects may be thrown as exceptions in C++.

b. "LINE D" gives a compile time error because it calls a function that throws an exception outside of a `try/catch` block.

c. When "LINE D" executes an exception is thrown by `hurl` which is not caught, causing the program to terminate.

d. When "LINE D" executes an exception is thrown by `hurl` which is not caught, causing the program to terminate after executing "LINE E".

e. **a** and **b**.