ECE 30862 Fall 2017, Third Exam

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

THE LAST PAGE IS THE ANSWER SHEET. TEAR IT OFF AND PUT ALL ANSWERS THERE. TURN IN BOTH PARTS OF THE TEST WHEN FINISHED.

You have until 9:00PM to take this exam. 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 this sheet, a largely blank page, 12 pages with 40 questions, and the answer sheet. As soon as the test begins, check that your exam is complete and *let Prof. Midkiff know immediately if it does 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.

For each question that is a comment on a line, answer, on the answer sheet, what is printed by the commented line. If a runtime or compile time error occurs, answer Err. If the statement is legal and nothing is printed, answer Ok. If the statement causes an error, execute the remainder of the program as if the erroneous statement did not exist.

Check the front board occasionally for corrections and before asking a question.

Every question is worth 2.5 points.

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:

This page intentionally left blank.

```
class Base {
public:
   virtual void print( );
};
```

Base.cpp

```
void Base::print( ) {
  std::cout << "B" << std::endl;
}
```

```
class A : public Base {
public:
   virtual void print( );
};
```

A.cpp

```
using namespace std;

void A::print( ) {cout << "A" << endl;}
```

```
class Int {
public:
   virtual void print( );
};
```

Int.cpp

```
using namespace std;

void Int::print( ) {
   std::cout << "I" << std::endl;
}
```

Q2: In the constructor, called from the line marked C in `main`, what is the type of a3 in the call from main.cpp?

Q3: In the line marked A, is the assignment legal?

Q4: In the line marked B, is the assignment legal?

```
template <class T1, class T2, class T3> class
Node {
private:
      T1  t1;
      T2  t2;
      T3  t3;
public:
   Node(T1, T2, T3);
   virtual ~Node( );
   void foo(T1, T2, T3);
   void print( );
};

template <class T1, class T2, class T3>
Node<T1,T2,T3>::Node(T1 a1, T2 a2, T3 a3) :
    t1(a1), t2(a2), t3(a3) { }

template <class T1, class T2, class T3>
Node<T1,T2,T3>::~Node( ) { }

template <class T1, class T2, class T3>
void Node<T1,T2,T3>::foo(T1 a1, T2 a2, T3 a3) {
   a1 = a2;  // A
   a2 = a1;  // B
}

template <class T1, class T2, class T3>
void Node<T1,T2,T3>::print( ) {
   t1->print( );
   t2->print( );
   t3.print( );
}

int main( ) {

   Base* b = new Base( );
   A* a = new A( );
   Int i;

   Node<Base*, A*, Int> n(b, a, i); // C

   n.print( ); // Q1
}
```

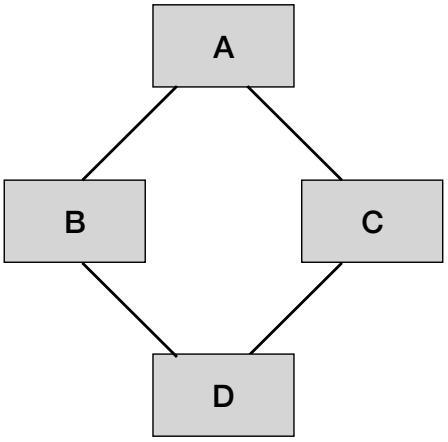

Figure 1

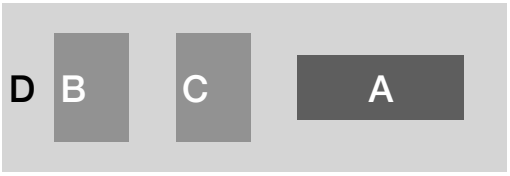

Figure 2

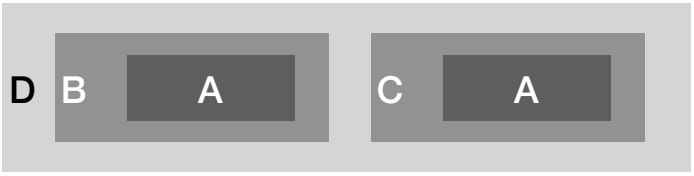

Figure 3

```
class A { };

class B : public A { };

class C : public A { };

class D : public B, C { };

int main( ) {

   C* c = new C( ); // Line A
   D* d = new D( ); // Line B
}
```

Program 1

```
class A { };

class B : public virtual A { };

class C : public virtual A { };

class D : public B, C { };

int main( ) {

   C* c = new C( ); // Line A
   D* d = new D( ); // Line B
}
```

Program 2

C++ Question. Figure 1 shows the inheritance Diagram for both Program 1 and Program 2.

Q5: Does Figure 2 represent a D object in Program 1, Program 2, or neither?
Q6: Does Figure 3 represent a D object in Program 1, Program 2, or neither?
Q7: Who calls the A constructor in Program 1?
(a) the B and C constructors
(b) the D constructor
(c) none of the above

Q8: Who calls the A constructor in program 2?
(a) the B and C constructors
(b) the D constructor
(c) none of the above

```
class A {
public:
  A( );
  A(int);
};

// A.cpp
A::A( ) {std::cout << "A" << std::endl;}
A::A(int i) {std::cout << "Ai" << std::endl;}

class B : public A {
public:
  B( );
};

// B.cpp
B::B( ) {std::cout << "B" << std::endl;};

class C : public B {
public:
  C( );
};

// C.cpp
C::C( ) {std::cout << "C" << std::endl;};
```

```
// main.cpp
void foo(char c, double d) {
  std::cout << "cd" << std::endl;
}

void foo(int s, double d) {
  std::cout << "sd" << std::endl;
}

void foo(double d, int i) {
  std::cout << "di" << std::endl;
}

void foo(double d, short s) {
  std::cout << "ds" << std::endl;
}

void foo(A a1, A a2) {
  std::cout << "aa" << std::endl;
}

void foo(const A& a, const C& c) {
  std::cout << "ac" << std::endl;
}

int main( ) {
  A a;
  B b;
  C co;
  int i;
  float f;
  double d;
  char c;

  foo(i,i); // Q9
  foo(i,f); // Q10
  foo(b, b); // Q11
}
```

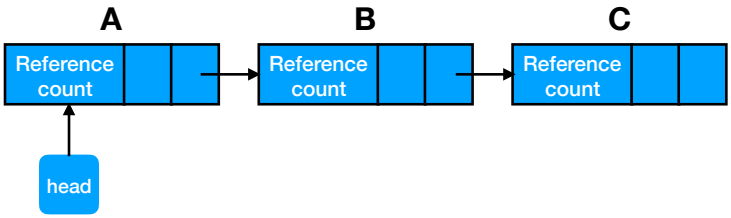

Figure 1

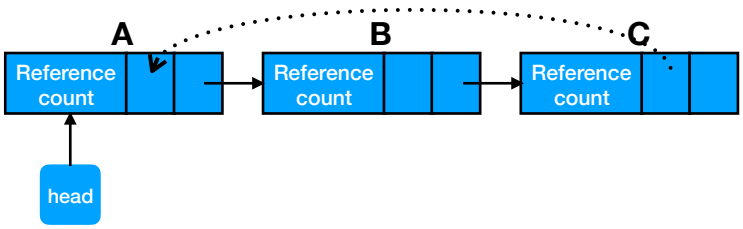

Figure 2

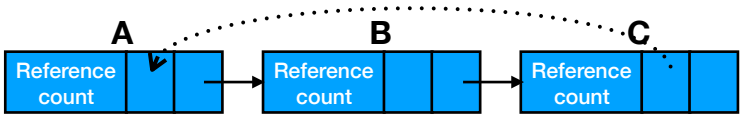

Figure 3

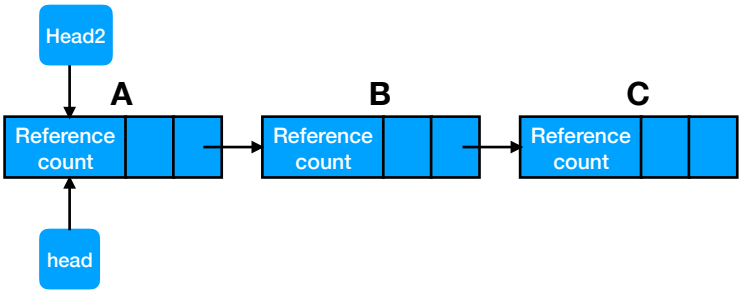

Figure 4

C++ questions. For Q12 - Q15, solid lines are C++ shared smart pointers and dotted lines are weak pointers.

Q12: What are the reference counts for nodes A, B and C in Figure 1? (give the numbers)
Q13: What are the reference counts for nodes A, B and C in Figure 2? (give the numbers)
Q14: What are the reference counts for nodes A, B and C in Figure 3? (give the numbers)
Q15: What are the reference counts for nodes A, B and C in Figure 4? (give the numbers)

```
class A {
public:
   A( );
   void f0( );
   virtual void f1( );
   virtual void f1(int);
   virtual void f2(float);
   virtual void f3(A*);
private:
   virtual void f4( ); // define in derived,
                       // call from passed in.
};

// A.cpp
A::A( ) {std::cout << "A" << std::endl;}

void A::f1( ) {std::cout << "A1v" << std::endl;}

void A::f1(int i) {std::cout << "A1i" << std::endl;}

void A::f2(float f) {std::cout << "A2f" << std::endl;}

void A::f3(A* a) {
   std::cout << "A3A" << std::endl; a->f4( ); a->f1(2);
}

void A::f4( ) {std::cout << "A4" << std::endl;}

class B : public A {
public:
   B( );
   virtual void f1(char);
   virtual void f2( );
   virtual void f3(B*);
   virtual void f4( );
};

// B.cpp
B::B( ) {std::cout << "B" << std::endl;};

void B::f1(char c) {std::cout << "B1c" << std::endl;}

void B::f2( ) {std::cout << "B2" << std::endl;}

void B::f3(B* b) {std::cout << "B3b" << std::endl;}

void B::f4( ) {std::cout << "B4" << std::endl;}
```

```
class C : public B {
public:
   C( );
   void f4( );
};

// C.cpp
C::C( ) { }
void C::f4( ) {
   std::cout << "C4v" << std::endl;
}

int main( ) {
   A* aP;
   B* bP = new B( );
   C* cP = new C( );

   bP->f1('a'); // Q16
   bP->f1(1); // Q17
   bP->f2( ); // Q18
   bP->f3(bP); // Q19
   bP->f3(cP); // Q20

   aP = bP;
   A aP->f4( ); // Q21
}
```

Page 6. **Java questions.**

Q22: A Java program has a large data structure referenced to by a static variable `r`. What can be done to let the memory used by the large data structure be reallocated, if needed?
(a) call `delete` on `r`, i.e, "`delete r`"
(b) call `free` on `r`, i.e, "`free r`".
(c) set `r` equal to `null`.
(d) invoke the garbage collector, and only add the code below to the program:

```
Runtime r = Runtime.getRuntime();
r.gc();
```

(e) none of the above.

Q23: Garbage collection is good because:
(a) Heap allocated objects are only deleted when they are not reachable, eliminating dangling pointer issues;
(b) It saves the programmer from having to manually manage memory;
(c) It reduces the chance of having memory leaks;
(d) It reduces or eliminates bugs arising from memory management;
(e) All of the above;
(f) None of the above.

Q24: When the garbage collector runs
(a) All unreachable objects are collected on every run of the GC;
(b) At least some unreachable objects are collected if they exist;
(c) There is some overhead for garbage collection.
(d) a and c;
(e) b and c.

Page 7. **Java questions.**

```
class T1 implements Runnable {

  static Object obj = new Object( );
  static int count = 0;

  public void run( ) {
    for (int i = 0; i < 10000000; i++) {
      count++;
    }
  }
}

class T2 implements Runnable {

  public void run( ) {
    for (int i = 0; i < 10000000; i++) {
      T1.count--;
    }
  }
}

class Main {

  public static void main(String args[]) throws InterruptedException {
    Thread t1 = new Thread(new T1( ));
    Thread t2 = new Thread(new T2( ));

    t1.start( );
    t2.start( );
    t1.join( );
    t2.join( );
    System.out.println(T1.count); // A

    T1.count = 0;
    t1.run( );
    t2.run( );
    t1.join( );
    t2.join( );
    System.out.println(T1.count); // B
  }
}
```

Q25. In the line marked A, is the value of T1.count printed:

(a) 0
(b) Less than 0
(c) Greater than 0
(d) It can be 0, less than 0 or greater than 0.

Q26. In the line marked B, is the value of T1.count printed:

(a) 0
(b) Less than 0
(c) Greater than 0
(d) It can be 0, less than 0 or greater than 0.

The difference between this code and the code on the previous page is the *synchronized statements (shown in **bold italics** in T1 and T2.)*

```
class T1 implements Runnable {

  static Object obj = new Object( );
  static int count = 0;

  public void run( ) {
    for (int i = 0; i < 10000000; i++) {
      synchronized(obj) {
        count++;
      }
    }
  }
}
class T2 implements Runnable {

  public void run( ) {
    for (int i = 0; i < 10000000; i++) {
      synchronized(T1.obj) {
        T1.count--;
      }
    }
  }
}
class Main {

   public static void main(String args[]) throws InterruptedException {
   Thread t1 = new Thread(new T1( ));
   Thread t2 = new Thread(new T2( ));

   t1.start( );
   t2.start( );
   t1.join( );
   t2.join( );
   System.out.println(T1.count); // A
  }
}
```

Q27. In the line marked A, what is the value of T1.count that is printed:

(a) 0
(b) Less than 0
(c) Greater than 0
(d) It can be 0, less than 0 or greater than 0.

The difference between this code and the code on the previous page is *shown in **bold italics** in T1 and T2.*

```
class T1 implements Runnable {

  static Object obj = new Object( );
  static int count = 0;

  public void run( ) {
    for (int i = 0; i < 10000000; i++) {
      synchronized(obj) {
        count++;
      }
    }
  }
}

class T2 implements Runnable {

  static Object obj = new Object( );

  public void run( ) {
    for (int i = 0; i < 10000000; i++) {
      synchronized(obj) {
        T1.count--;
      }
    }
  }
}

class Main {

  public static void main(String args[]) throws
InterruptedException {
    Thread t1 = new Thread(new T1( ));
    Thread t2 = new Thread(new T2( ));

    t1.start( );
    t2.start( );
    t1.join( );
    t2.join( );
    System.out.println(T1.count); // A
  }
}
```

Q28. In the line marked A, is the value of T1.count printed:

(a) 0
(b) Less than 0
(c) Greater than 0
(d) It can be 0, less than 0 or greater than 0.

```
class T1 implements Runnable {

  static int count = 0;

  private synchronized void counter( ) {
    for (int i = 0; i < 10000000; i++) {
      count++;
    }
  }

  public void run( ) {
    counter( );
  }
}

class T2 implements Runnable {

  private synchronized void counter( ) {
    for (int i = 0; i < 10000000; i++) {
      T1.count--;
    }
  }

  public void run( ) {
    counter( );
  }
}

class Main {

  public static void main(String args[]) throws
InterruptedException {
    Thread t1 = new Thread(new T1( ));
    Thread t2 = new Thread(new T2( ));

    t1.start( );
    t2.start( );
    t1.join( );
    t2.join( );
    System.out.println(T1.count); // A
  }
}
```

Q29. In the line marked A, is the value of T1.count printed:

(a) 0
(b) Less than 0
(c) Greater than 0
(d) It can be 0, less than 0 or greater than 0.

```
class T1 implements Cloneable {
  static int[ ] siv = {0, 1, 2};
  int[ ] iv;

  T1(int i) {
    iv = new int[2];
    for (int j = i; j < i+2; j++) {
      iv[j-i] = j;
    }
  }

  public Object clone( )
    throws CloneNotSupportedException {
    super.clone( );
    T1 t = new T1(0);
    siv = new int[2];
    t.iv = new int[2];
    for (int i = 0; i < siv.length; i++) {
      siv[i] = i;
      t.iv[i] = 2*i;
    }
    return t;
  }

  public String toString( ) {
      return " "+siv[0]+", "+iv[0];
  }
}
```

```
class Main {

  public static void main(String args[])
      throws Exception {
    T1 t1 = new T1(10);
    T1 t2 = new T1(100);

    System.out.println(t1); // Q30
    System.out.println(t2); // Q31

    t2 = t1;
    t1.siv[0] = 10;
    t1.iv[0] = 90;
    System.out.println(t1); // Q32
    System.out.println(t2); // Q33

    t2 = (T1) t1.clone( );
    t1.siv[0] = 20;
    t1.iv[0] = 20;
    System.out.println(t1); // Q34
    System.out.println(t2); // Q35
  }
}
```

```
class T1 {
  private void f1( ) {
    System.out.println("T1");
  }
;
  public void f2( ) {
    System.out.println("T1");
  }

  public void f3(int i) {
    System.out.println("T1");
  }

  public void f5( ) {
    f1( ); f2( );
  }
}

class T2 extends T1 {

  public void f1( ) {
    System.out.println("T2");
  }

  public void f2( ) {
    System.out.println("T2");
  }

  public void f3(short s) {
    System.out.println("T2");
  }

  public static void f4( ) {
    System.out.println("T2");
  }
}
```

```
class Main {

  public static void main(String args[])
      throws Exception {
    T1 t1 = new T1( );
    T2 t2 = new T2( );
    short s = (short) 1;

    t1.f1( ); // Q36

    t1 = t2;
    t1.f1( ); // Q37
    t1.f3(1); // Q38
    t1.f3((short) 1); // Q39
    t1.f4( ); // Q40
  }
}
```

ECE 30862 Fall 2016 Third Exam Answer Sheet

Name (Printed):

Name (Signed):

1.	**21.**
2.	**22.**
3.	**23.**
4.	**24.**
5.	**25.**
6.	**26.**
7.	**27.**
8.	**28.**
9.	**29.**
10.	**30.**
11.	**31.**
12.	**32.**
13.	**33.**
14.	**34.**
15.	**35.**
16.	**36.**
17.	**37.**
18.	**38.**
19.	**34.**
20.	**40.**