ECE 30862 Fall 2019 Second Exam Answer Sheet

Put your name above!

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.	**39.**
20.	**40.**

This page intentionally left almost blank

ECE 30862 Fall 2019, Test 2

DO NOT START WORKING ON THIS UNTIL TOLD TO DO SO.

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

You have until 7:30 to take this exam. The total number of points should be 100, Each of the 40 questions is worth 2.5 points. After taking the test turn in both the test and the answer sheet.

Your exam should have 10 (ten) pages total (including this cover page, the answer sheet and one almost entire blank page). 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 answer sheet whatever assumptions you made to answer the question, and answer it under those assumptions.

Check the front board occasionally for corrections.

Programs may be given without "#include" statements for brevity. Assume all needed includes are present. "std::endl" has been left off for brevity. You may use newlines in your answer, or not, without affecting its correctness.

Each page with questions, except for page 4, has the following instructions:

> The code on this page and the facing page are used for questions x - y. If something is printed on a line that is a question (has a Qx comment, where "x" is a natural number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist for the rest of the program. If the statement prints nothing but is correct, answer "Ok". If a value is uninitialized, answer "uninit" or something similar.

where "x" and "y" are question numbers, so you don't need to read these if you are running short on time. Page 4 has slightly different instructions.

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

C++ questions. The code on this page is for questions 1 - 6. All answers will be "Ok" or "Err". Answer "Err" if the statement gives an error at compile or run time, "Ok" otherwise.

```
// A.h
class A {
public:

   int v;
    A( );
    virtual ~A( );
};
```

```
// A.cpp
A::A( ) {v=1;}
A::~A( ) { }
```

```
// B.h
class B : public A {
public:
   virtual void b( );
};
```

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

```
// C.h
class C : public A {
public:
   virtual void c( );
};
```

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

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

   A* aP1 = new A( );
   A* aP2 = new A( );
   B* bP1 = new B( );
   C* cP1 = new C( );
   aP1 = bP1; // Q1
   cP1 = aP2; // Q2
   cP1 = (C*) aP1; // Q3
   cP1 = static_cast<C*>(aP1); // Q4
   cP1 = dynamic_cast<C*>(aP2); // Q5
   i = cP1->v; // Q6
}
```

The code on this page is used for questions 7 - 12. If something is printed on a line that is a question (has a Qx comment, where "x" is a natural number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist for the rest of the program. If the statement prints nothing but is correct, answer "Ok". If a value is uninitialized, answer "uninit" or something similar.

```
// Wierdmath.h
class Wierdmath {
private:
    int val;

public:
   Wierdmath(int);
   ~Wierdmath( );

   int getValue( ) const;

   Wierdmath operator+(const Wierdmath&);
   std::ostream& operator+(std::ostream&);
   Wierdmath operator-( );
   friend Wierdmath operator-(
      const Wierdmath&, const Wierdmath&);
   friend std::ostream& operator<<(
      std::ostream&, const Wierdmath&);
};

// Wierdmath.cpp
Wierdmath::Wierdmath(int v) : val(v) { }
Wierdmath::~Wierdmath( ) { }

int Wierdmath::getValue( ) const {
   return val;
}

Wierdmath Wierdmath::operator+(
   const Wierdmath& x) {
   return Wierdmath(val - x.val);
}

std::ostream& Wierdmath::operator+(
   std::ostream& os) {
   os << "#" << val << "#";
   return os;
}

Wierdmath Wierdmath::operator-( ) {
   return Wierdmath(-val);
}

Wierdmath operator-(const Wierdmath& w,
                    const Wierdmath& z) {
   return Wierdmath(w.val + z.val);
}

std::ostream& operator<<(std::ostream& os,
                         const Wierdmath& w) {
   os << "(" << w.getValue( ) << ")";
   return os;
}

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

   Wierdmath w0(0);
   Wierdmath w1(1);
   Wierdmath w2(2);
   Wierdmath w3(3);

   std::cout << w0 << std::endl; // Q7

   w0 = w2 + w1;
   std::cout << w0 << std::endl; // Q8

   w0 = w3 + w2;
   std::cout << w2 << std::endl; // Q9

   w2 + std::cout; // Q10
}
```

Q11: would function `Wierdmath operator-(const Wierdmath& w, const Wierdmath& z)` compile without errors if it were not a friend function?
Q12: would function `std::ostream& operator<<(std::ostream& os, const Wierdmath& w);` compile without errors if it were not a friend function? It would still be a free function, i.e., not a member function.

The code on this page and the facing page are used for questions 13 - 18. If something is printed on a line that is a question (has a Qx comment, where "x" is a natural number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist for the rest of the program. If the statement prints nothing but is correct, answer "Ok". If a value is uninitialized, answer "uninit" or something similar.

```
// Vec.h
class Vec {
public:
   int* values;
   int ub;
   Vec(int);
   Vec(const Vec&);
   virtual ~Vec( );

   virtual void incValue(int);

   virtual Vec operator=(const Vec&);
};

// Vec.cpp
Vec::Vec(int n) {
   ub = n;
   values = new int[n];
   for (int i = 0; i < n; i++) {
      values[i] = i;
   }
}

Vec::Vec(const Vec& v) {
   values = new int[v.ub];
   ub = v.ub;
   for (int i = 0; i < v.ub; i++) {
      values[i] = values[i]+1;
   }
}
```

```
Vec Vec::operator=(const Vec& v) {
   if (this != &v) {
      delete values;
      values = new int[v.ub];
      ub = v.ub;
      for (int i = 0; i < ub; i++) {
         values[i] = values[i]+2;
      }
   }
   return *this;
}

Vec::~Vec( ) {
   delete values;
}

void Vec::incValue(int i) {
   values[i]++;
}
```

```
// main.cpp
void negate1 (Vec v) {
   v.values[0] = -v.values[1];
}

void negate2 (Vec& v) {
   v.values[0] = -v.values[1];
}

void negate (Vec* v) {
   v->values[0] = -v->values[0];
}

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

   Vec v1(2);
   Vec v2 (2);
   Vec& v1R = v1;



   v1.incValue(0);
   std::cout << v1.values[0] << " " << v1R.values[0] << std::endl; // Q13

   negate1(v1);
   std::cout << v1.values[1] << std::endl; // Q14

   negate2(v1);
   std::cout << v1R.values[1] << std::endl; // Q15

   v1.values[1] = -10;
   std::cout << v1R.values[1] << std::endl; // Q16

   v1 = v2;
   v1.values[1] = -10;
   std::cout << v1.values[1] << " " << v2.values[1] << std::endl;   // Q17

   v1R = v2;
   v1.values[1] = -10;
   std::cout << v1.values[1] << " " << v1R.values[1] << " " << v2.values[1] << std::endl;  // Q18
}
```

The code on this page and the facing page are used for questions 19 - 33. If something is printed on a line that is a question (has a Qx comment, where "x" is a natural number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist for the rest of the program. If the statement prints nothing but is correct, answer "Ok". If a value is uninitialized, answer "uninit" or something similar.

```
public class Base {
   private void foo( ) {
      System.out.println("B:foo");
   }

   public Base( ) {
      System.out.println("Base");
   }

   public Base(int i) {
      System.out.println("Base(i)");
   }

   public void bar(int i) {
      System.out.println("bar(int)");
      foo( );
   }

   public void bar(Base b) {
      System.out.println("bar(Base)");
      b.foo( );
   }

   public void bar(long l) {
      System.out.println("bar(long)");
      foo( );
   }

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

   public void f2( ) {
      System.out.println("Base:f2");
   }
}
```

```
public class Derived extends Base {

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

   public static int si;

   public Derived( ) {
      System.out.println("Derived");
      si = 0;
   }

   public Derived(int i) {
      super(i);
      System.out.println("Derived(i)");
      si = i;
   }

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

   public void f3( ) {
      System.out.println("Der:f3");
   }
}
```

```
class Main {

   public static void main(String args[]) {

      Base b = new Base( );
      Derived d = new Derived( ); // Q19
      Derived d2 = new Derived(1); // Q20
      short s = 10;

      System.out.println(d.si + " " + d2.si); // Q21

      b.foo( );  // Q22
      b.bar(s);  // Q23
      b.bar(d2);  // Q24
      b.f1( ); // Q25
      b = d;
      b.f1( ); // Q26
      b.f2( ); // Q27
      b.f3( ); // Q28

      d.foo( );  // Q29
      d.f1( ); // Q30
      d.f2( ); // Q31
      d.f3( ); // Q32
      d = b; // Q33
   }
}
```

The code on this page and the facing page are used for questions 34 - 40. If something is printed on a line that is a question (has a Qx comment, where "x" is a natural number) say what is printed. If the line has an error at either compile or runtime, answer "Err" and assume the statement doesn't exist for the rest of the program. If the statement prints nothing but is correct, answer "Ok". If a value is uninitialized, answer "uninit" or something similar.

```
public abstract class Base {
   abstract void foo( );
   abstract void bar( );
}

public class Derived extends Base implements I1, I2 {

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

}

interface I1 {
   int constI = 4;

   void foo(  );

   void f1( );
}

interface I2 {
   int constI = 5;

   void foo(  );
}

class Main {

   public static void main(String args[]) throws Exception {

      Base b = new Base( ); // Q34
      I1 i1 = new I1( ); // Q35
      I1 i1 = new Derived( ); // Q36
      Base b = new Derived( );
      b.foo( ); // Q37
      b.f1( ); // Q38

      Derived d = new Derived( ); // Q39
      System.out.println(d.constI); // Q40
   }
}
```