ECE 30862 Fall 2016, Second 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 7:30PM 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, 10 pages with 50 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. *Check the front board occasionally for corrections.*

Every question is worth 2 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 (printed, worth 1 pt):

Last four digits of your ID:

This page intentionally left almost blank

For each statement below which has a question number (e.g., **Q7**), write “Err” if the access is illegal and “OK” if it is legal. There is not need to say what is printed.

```
class Base { // Base.h
public:
   int i, j, l;
protected:
   int k;
public:
   Base( );
   virtual ~Base( );
};

Base::Base( ) {  } // Base.cpp

Base::~Base( ) { }

class D1 : protected Base { // D1.h
public:
   int i, j;
   D1( );
   virtual ~D1( );
};

D1::D1( ) { } // D1.cpp
D1::~D1( ){ }

class D2 : public D1 { // D2.h
public:
   D2( );
   virtual ~D2( );
   void print( );
};

D2::D2( ) { } // D2.cpp
D2::~D2( ){ }
void D2::print( ) {
   cout << i; // Q1
}
```

```
#include "Base.h"
#include "D1.h"
#include "D2.h"
#include <iostream>
#include <string>
using namespace std;

int main(void) {

   Base* b = new Base( );
   Base* d1 = new D1( );
   Base* d2 = new D2( );

   cout << d1->i; // Q2
   cout << d1->k; // Q3
   cout << d1->l; // Q4

   cout << d2->i; // Q5
   cout << d2->k; // Q6
   cout << d2->l; // Q7

   b = d1;
   cout << b->i; // Q8
   cout << b->j; // Q9
   cout << b->l; // Q10
}
```

For each statement below which has a question number (e.g., **Q11**), write the output that results from executing the statement on the answer sheet. All statements are legal.

```
class B { // B.h
public:
  B( );
  virtual ~B( );
  virtual void f1( );
  void f2( );
  void f3( );
};

B::B( ) {  } // B.cpp

B::~B( ) { }

void B::f1( ) {cout << "B::f1" << endl;}

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

void B::f3( ) {cout << "B::f3" << endl;}

class C : public B { // C.h
public:
  C( );
  C(int );
  virtual ~C( );
  void f1( );
  virtual void f2( );
  void f3( );
};

C::C( ) {  }
C::C(int i) {  }

C::~C( ) { }

void C::f1( ) {cout << "C::f1" << endl;}
void C::f2( ) {cout << "C::f2" << endl;}
void C::f3( ) {cout << "C::f3" << endl;}
```

```
class D : public C {  // D.h
public:
  D(int);
  D( );
  virtual ~D( );
  void f1( );
  void f2( );
  virtual void f3( );
};

D::D(int i) {  } // D.cpp
D::D( ) {  }

D::~D( ) { }

void D::f1( ) {cout << "D::f1" << endl;}
void D::f2( ) {cout << "D::f2" << endl;}
void D::f3( ) {cout << "D::f3" << endl;}

int main(void) { // main.cpp

  C c1(1);
  D d1(1);

  C c2 = d1;
  c2.f2( ); // Q11
  c2.f3( ); // Q12

  B& bR = (B&) c1;
  bR.f1( ); // Q13
  bR.f3( ); // Q14

  C& cR = (C&) d1;
  cR.f1( ); // Q15
  cR.f2( ); // Q16

  B* bP = &c1;
  bP->f2( ); // Q17
  bP->f3( ); // Q18

  C* cP = &d1;
  cP->f1( ); // Q19
  cP->f3( ); // Q20
}
```

For each statement below which has a question number (e.g., **Q21**), write the output that results from executing the statement on the answer sheet. All statements are legal.

```
class B { // B.h
public:
  B( );
  B(int);
  virtual ~B( );
  virtual void f1(int);
  virtual void f1(double);
};

B::B( ) {  } // B.cpp
B::B(int i) {  }

B::~B( ) { }

void B::f1(int i) {
  cout << "B::int" << endl;
};

void B::f1(double) {
  cout << "B::double" << endl;
};

class C : public B { // C.h
public:
  C( );
  C(int );
  virtual ~C( );
  void f1(int);
};

C::C( ) {  } // C.cpp
C::C(int i) {  }

C::~C( ) { }

void C::f1(int) {
  cout << "C::int" << endl;
};
```

```
int main(void) {

  B b1(1);
  C c1(1);
  int i = 1;
  double d = 1.0;

  b1.f1(i); // Q21
  b1.f1(d); // Q22
  c1.f1(d); // Q23

  B* bP = &c1;
  bP->f1(d); // Q24
}
```

For each statement below which has a question number (e.g., **Q25**), write the output that results from executing the statement on the answer sheet. All statements are legal.

```
class B { // B.h
public:
  int i;
  B(int);
  virtual ~B( );
};

B::B(int j) : i(j) {  } // B.cpp

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

```
void f1(B b) { // main.cpp
  b.i = 0;
};

void f2(B& b) {
  b.i = 0;
};

void f3(B* b) {
  b->i = 0;
};

int main(void) {

  B b(4);
  B& bR = b;

  f1(b);
  cout << b.i << endl; // Q25
  b.i = 4;

  f1(bR);
  cout << bR.i << endl; // Q26
  bR.i = 4;

  f2(b);
  cout << b.i << endl; // Q27
  b.i = 4;

  f2(bR);
  cout << bR.i << endl; // Q28

  bR.i = 5;
  cout << b.i << endl; // Q29
}
```

For each statement that is a question, what is printed by constructors or destructors when the statement executes. For **Q30**, what is printed when *foo* is called? For **Q33**, what is printed by the *cout* statement when *foo* is called. All statements are legal.

```
class B {
public:
  int i;
  B( );
  B(int);
  B(B&);
  virtual ~B( );
};

B::B( ) {cout << "B" << endl; i = 0;}
B::B(int j) : i(j) {cout << "B(int)" << endl;}
B::B(B& b) {
  cout << "B(&B b)" << endl;
  this->i=-b.i;}

B::~B( ) {cout << "~B" << endl;}

class C : public B {
public:
  C( );
  C(int );
  virtual ~C( );
};

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

C::~C( ) {cout << "~C" << endl;}
```

```
#include "B.h"
#include "C.h"
#include <iostream>
#include <string>
using namespace std;

void foo(B par) { // Q30
   cout << par.i << endl;
}

int main(void) {

  B b1(1); // Q31
  C c1(1); // Q32
  foo(b1); // Q33
}
```

Q34: what, if anything, is anything printed after the call to *foo* when *b1* and *c1* are popped off the stack?

Answer the questions below using the code below. All statements are legal.

```
class Weird { // Weird.h
private:
  int i;
public:
  Weird( );
  Weird(int);
  virtual ~Weird( );

  Weird operator+(Weird);
  Weird getI( );

  friend Weird operator*(Weird, Weird);
  friend ostream& operator<<(ostream& os, const
Weird&);
};

Weird::Weird( ) {i=0;}; // Weird.cpp
Weird::Weird(int j) : i(j) { }
Weird::~Weird() { }

Weird Weird::operator+(Weird w) {
  int res = this->i * w.i;
  return Weird(res);
}

Weird operator*(Weird w1, Weird w2) {
  int res = w1.i + w2.i;
  return Weird(res);
}

ostream& operator<<(ostream& os, const Weird& w) {
  return os << w.i;
}
```

```
#include "Weird.h"
#include <iostream>
using namespace std;

int main(void) {

  Weird w1(3);
  Weird w2(5);
  Weird w3(7);

  cout << w1+w2 << endl; // LINE A
  cout << w1+w2*w3 << endl; // LINE B
}
```

Q35: If the overloaded * was declared in the Weird class, how many parameters need to be specified by the programmer?

Q36: In LINE A, which of w1 and w2 is passed as the *this* pointer to the function?

Q37: What is printed by **LINE A**?

Q38: What is printed by **LINE B**

Q39: Could the overloaded **<<** operator be declared as a member function of the Weird class? Answer “yes” or “no”.

Answer the questions below using the code below. All statements are legal.

```
class Exp { // Exp.h
public:
  Exp( );
  virtual ~Exp( );
  string msg( );
};

Exp::Exp( ) { } // Exp.cpp
Exp::~Exp( ) { }
string Exp::msg( ) {return "E1";}

class Exp2 { // Exp2.h
public:
  Exp2( );
  virtual ~Exp2( );
  string msg( );
};

Exp2::Exp2( ) { } // Exp2.cpp
Exp2::~Exp2( ) { }
```

```
void foo( ) { // main.cpp
  throw 1.0;
}

void heave(int i) {
  if (i == 0) throw Exp( );
  if (i == 1) throw 2;
  if (i == 2) throw Exp2( );
  foo( );
}

int main( ) {
  for (int i = 0; i < 3; i++) {
    try {
      heave(i);
    } catch (int i) {
      cout << "caught it " << i << endl;
    } catch (Exp e) {
      cout << e.msg( ) << endl;
    }
  }
  return 0;
```

Q40: What is printed in the try-catch clause when *i = 0*?
Q41: What is printed in the try-catch clause when *i = 1*?
Q42: Would declaring void *heave(int i)* as *void heave(int i) throw(int, Exp, Exp2)* guarantee that only these exceptions are thrown? Answer “yes” or “no”.
Q43: Would it be legal to add a finally clause to the try-catch? Answer “yes” or “no”.

Answer the questions using the code, which is a template description, below. The code is legal.

```
template <class T1, class T2> class Tuple {
  private:
    T1 v1;
    T2 v2;
public:
  Tuple(T1, T2);
  virtual ~Tuple( );
  void print( );
};

template <class T1, class T2>
Tuple<T1,T2>::Tuple(T1 a1, T2 a2) : v1(a1), v2(a2) { }

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

template <class T1, class T2>
void Tuple<T1,T2>::print( ) {
  cout << v1 << ", " << v2 << endl;
}
```

Q44: If a program uses this template by specifying code like Tuple<int, String> in our program, what is the type of v1?

Q45: If a program uses this template by specifying code like Tuple<int, String> in our program, what is the type of v2?

Q46: Does T2 have to be the name of a class? Answer “yes” or “no”.

Answer the questions using the code below. The code is legal.

```
public class T1 extends Thread {

  public static int count=0;

  public T1( ) { }

  private void update( ) {
    int v = count;
    try {
      sleep(10);
    } catch (Exception e) { }
    v++;
    count = v;
  }

  public void run( ) {
    for (int i = 0; i < 1000; i++) {
      update( );
    }
  }
}

public class T2 extends Thread {

  public static int count=0;

  public T2( ) { }

  private synchronized void update( ) {
    int v = count;
    try {
      sleep(10);
    } catch (Exception e) { }
    v++;
    count = v;
  }

  public void run( ) {
    for (int i = 0; i < 1000; i++) {
      update( );
    }
```

```
class Main {

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

    T1 t1_1 = new T1( );
    T1 t1_2 = new T1( );
    t1_1.start( );  t1_2.start( );

    t1_1.join( ); t1_2.join( );
    System.out.println("T1.start, "+T1.count); // LINE A

    T1.count = 0;
    t1_1.run( ); t1_2.run( );
    System.out.println("T1.run, "+T1.count); // LINE B

    T2 t2_1 = new T2( );
    T2 t2_2 = new T2( );
    t2_1.start( ); t2_2.start( );
    t2_1.join( ); t2_2.join( );
    System.out.println("T2, "+T2.count); // LINE C
  }
}
```

Q47: Answer which is most true of what is printed by LINE A:

(a) Exactly 2000
(b) A value that is greater than or equal to 0 and less than or equal to 2000
(c) Any value is possible to be printed

Q48: Answer which is most true of what is printed by LINE B:

(a) Exactly 2000
(b) A value that is greater than or equal to 0 and less than or equal to 2000
(c) Any value is possible to be printed

Q49: Answer which is most true of what is printed by LINE C:

(a) Exactly 2000
(b) A value that is greater than or equal to 0 and less than or equal to 2000
(c) Any value is possible to be printed.

Answer the questions using the code below. T3 is the same as T2 except for the bold code in T3's update function. The code is legal.

```
public class T3 extends Thread {

  public static int count=0;
  private static Object o = new Object( );

  public T3( ) { }

  private void update( ) {
    synchronized(o) {
      int v = count;
      try {
        sleep(10);
      } catch (java.lang.InterruptedException e) { }
      v++;
      count = v;
    }
  }

  public void run( ) {
    for (int i = 0; i < 1000; i++) {
      update( );
    }
  }
}
class Main {

  public static void main(String args[]) throws
java.lang.InterruptedException {

    T3 t3_1 = new T3( );
    T3 t3_2 = new T3( );
    t3_1.start( );
    t3_2.start( );

    t3_1.join( );
    t3_2.join( );
    System.out.println("T3, "+T3.count); // LINE A

  }
}
```

Q50: Answer which is most true of what is printed by LINE A:

(a) Exactly 2000.

(b) A value that is greater than or equal to 0 and less than or equal to 2000.

(c) Any value is possible to be printed.

All answers should be on this sheet. Put your name on the sheet.

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

16.

17.

18.

19.

20.

21.

22.

23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

34.

35.

36.

37.

38.

39.

40.

41.

42.

43.

44.

45.

46.

47.

48.

49.

50.