ECE 30862 Fall 2013 Final Exam Answer Sheet

All answers should be on this sheet which is printed on both sides. Both this sheet and your test must be signed and turned in. You may detach this sheet from the rest of the test to make it easier to write your answers on it. Each question is worth 2 points out of 106 points.

On my honor I have neither given nor received disallowed aid on this test.

Name (Printed):	Name (Signed):
1.	**14.**
2.	**15.**
3.	**16.**
4.	**17.**
5.	**18.**
6.	**19.**
7.	**20.**
8.	**21.**
9.	**22.**
10.	**23.**
11.	**24.**
12.	**25.**
13.	**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.

51.

52.

53.

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

Your exam should have 21 sheets total (including this cover sheet and the answer sheet.) *Please let Prof. Midkiff know immediately if it does not.* **ALL ANSWERS SHOULD BE ON THE ANSWER SHEET.**

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 the course.

Name (must be signed to be graded):

Name (printed, worth 1 pt):

Last four digits of your ID:

For the purpose of this question, the "%" (or modulo) operator gives the remainder. Thus 9%4 = 1. The actual semantics of the modulo operator are more subtle but don't make any difference for these questions.

```
#include <iostream>
using namespace std;
class Small {
public:
   int num;
   Small(int d) : num(d) { }
   Small(const Small& orig) {
      num = orig.num;
   }

   Small  operator*(const Small& d) {
      return Small((num*d.num)%5);
   }

  friend ostream& operator<<
         (ostream& os, const Small & arg);
};
```

```
Small operator/(const Small& d1,
                 const Small& d2) {
   return Small(d1.num/d2.num);
}

ostream& operator<< (ostream& os, const Small & arg) {
   os << arg.num;
   return os;
}

int main( ) {
   Small n1(3);
   Small n2(2);

   cout << -n1 << endl; // A
   cout << n1*n2 << endl; // B
   cout << n1*n1/n2 << endl; // C
   cout << n1/n2 << endl; // D
   return 0;
}
```

Question 1 (C++). Why is the << operator defined as a friend function of the Small class instead of a member function of the class?

a. It is accessible to functions that cannot access the Small class.

b. No real reason, it is just the preference of the programmer.

c. It cannot be defined as part of the Small class since the first argument is an ostream.

d. It is useful documentation to have both arguments shown explicitly.

Question 2 (C++). What is printed when statement A executes or when it is compiled?

a. 3

b. -3

c. Error because no unary minus operator is defined for Small objects.

f. A compile time error will result because arithmetic operators can never be applied to objects.

Question 3 (C++). What is printed when statement B executes or when it is compiled?

a. 0

b. 1

c. 2

d. 3

e. 4

f. Error because operator* takes either zero or one arguments and the multiply has two operands, n1 and n2.

h. A compile time error will result because we are trying to perform arithmetic on objects.

This code is the same as the code used by Questions 1 – 3.

```
#include <iostream>
using namespace std;
class Small {
public:
   int num;
   Small(int d) : num(d) { }
   Small(const Small& orig) {
      num = orig.num;
   }

   Small  operator*(const Small& d) {
      return Small((num*d.num)%5);
   }

  friend ostream& operator<<
         (ostream& os, const Small & arg);
};
```

```
Small operator/(const Small& d1,
                 const Small& d2) {
   return Small(d1.num/d2.num);
}

ostream& operator<< (ostream& os, const Small & arg) {
   os << arg.num;
   return os;
}

int main( ) {
   Small n1(3);
   Small n2(2);

   cout << -n1 << endl; // A
   cout << n1*n2 << endl; // B
   cout << n1*n1/n2 << endl; // C
   cout << n1/n2 << endl; // D
   return 0;
}
```

Question 4 (C++). What is printed when statement C executes or when it is compiled?

a. 0

b. 1

c. 2

d. 3

e. 4

f. Error because `operator*` takes either zero or one arguments and the multiply has two operands, `n1` and `n1`.

h. A compile time error will result because we are trying to perform arithmetic on objects.

```
public class Q6 extends Thread {

   public static int s1=0;
   public static int s2=0;

   public int f1;
   public int f2;

   Q6(int arg1, int arg2) {
      f1 = arg1;
      f2 = arg2;
   }

   public void run() {
      s1 = f1;
      s2 = f2;
   }
}
```

```
public static void main(String args[]) {
   Q6 t0 = new Q6(1,1);
   Q6 t1 = new Q6(2,2);

   t0.run( );
   t1.run( );

   System.out.println("s1: "+Q6.s1+", s2: "+Q6.s2);
}
```

Question 5 (Java Put the letter of the most correct answer on the answer sheet?

a. `t0.run( )` *must* finish executing before `t1.run( )` can begin

b. `t0.run( )` may finish executing before `t1.run( )` begins, but it doesn't have to.

Question 6 (Java

The main function in the program above is changed to:

```
public static void main(String args[]) {
   Q7 t0 = new Q7(1,1);
   Q7 t1 = new Q7(2,2);
    t0.start( ); // changed from run to start
    t1.start( ); // changed from run to start
    System.out.println("s1: "+Q7.s1+", s2: "+Q7.s2);
}
```

Put the letter of the most correct answer on the answer sheet?

a. `t0.start( )` *must* finish executing before `t1.start( )` can begin

b. `t0.start( )` may finish executing before `t1.start( )` begins, but it doesn't have to.

```
public class Q8 extends Thread {

   public static int s1=0;
   public static int s2=0;

   public int f1;
   public int f2;

   Q8(int arg1, int arg2) {
      f1 = arg1;
      f2 = arg2;
   }

   public synchronized void run() {
      s1 = f1;
      s2 = f2;
   }

   public static void main(String args[]) {
      Q8 t0 = new Q8(1,1);
      Q8 t1 = new Q8(2,2);

      t0.start( );
      t1.start( );

      System.out.println("s1: "+Q8.s1+", s2: "+Q8.s2);
   }
}
```

Question 7 (Java

Select the best answer and put it on your answer sheet.

a. `public synchronized void run( )` is not executed because it is never called in the program.

b. The `synchronized` in `public synchronized void run( )` synchronizes all object accesses in `run` and methods called from `run`.

c. The `synchronized` in `public synchronized void run( )` synchronizes all object accesses in `run` but not in any method called from `run`.

d. The `synchronized` in `public synchronized void run( )` acquires a lock on the object pointed to by the *this* pointer.

```
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS 5
void *PrintHello(void *threadid) {
   printf(\n%d: Hello World!\n", threadid);
   pthread_exit(NULL);
}
int main (int argc, char *argv[]){
   pthread_t threads[NUM_THREADS];
   int rc, t;
   for(t=0; t < NUM_THREADS; t++){
      printf("Creating thread %d\n", t);
      rc = pthread_create(&threads[t], NULL, PrintHello, (void *) &t);
      if (rc) {
         printf("ERROR; return code from pthread_create() is %d\n", rc);
         exit(-1);
      }
   }
   pthread_exit(NULL);
}
```

Question 8 (C++). Which statement below is most correct about the program above?

a. `pthread_create` creates a thread, but does not start it executing.

b. The `pthread_create` call creates a thread that executes the function "`PrintHello`".

c. The loop reuses one thread to execute "`PrintHello`" `NUM_THREADS` times.

d. The loop spawns `NUM_THREADS` threads, each of which executes "`PrintHello`".

e. (b) and (d).

Question 9 (C++). When `pthread_create(&threads[t], NULL, PrintHello, (void *) &t);` is called with `t = 1`, what is printed?

a. `Hello World!` 1

b. `Hello World!` 2

c. `Hello World!` 3

d. `Hello World!` 4

e. Any of a, b, c or d.

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

class X {
public:
   int j;
   X(int i) : j(i) { }
   X(const X& other) {
      if ( this != &other ) j = -other.j;
   }
   ~X() { }
};

int main() {
    X x1(2);
    X x2 = x1;
    cout << x1.j;
    cout << " " << x2.j << endl;
    return 0;
}
```

Question 10 (C++).
What is printed by this program?

a. `2, -2`

a. `2, 2`

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

class X {
public:
   int j;
   X(int i) : j(i) { }
   ~X() { }
};

int main() {
    X x1(2);
    X x2 = x1;
    cout << x1.j;
    cout << " " << x2.j << endl;
    return 0;
}
```

Question 11 (C++).
What is printed by this program?

a. `2, -2`

b. `2, 2`

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

class X {
public:
   X( ) {cout << "X constructed" << endl;}
   ~X( ) {cout << "X destroyed" << endl;}
};

class Y : public X {
public:
   Y( ) : X( ) {cout << "Y constructed" << endl;}
   ~Y( ) {cout << "Y destroyed" << endl;}
};

class Z : public Y {
public:
   Z( ) : Y( ) {cout << "Z constructed" << endl;}
   ~Z( ) {cout << "Z destroyed" << endl;}
};


int main()
{
    Z* z = new Z( );
    delete z;
}
```

Question 12 (C++).
What is printed by this program?

a.
```
X constructed
Y constructed
Z constructed
Z destroyed
Y destroyed
X destroyed
```

b.
```
Z constructed
Y constructed
X constructed
X destroyed
Y destroyed
Z destroyed
```

c.
```
Z constructed
Z destroyed
```

d.
```
Z constructed
Y constructed
X constructed
Z destroyed
Y destroyed
X destroyed
```

e.
```
X constructed
Y constructed
Z constructed
Z destroyed
Y destroyed
X destroyed
```

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

class Copied {
public:
   int* num;
   int len;

   Copied(int l) {
      num = new int[100];
      for (int i = 0; i < 100; i++) num[i] = i;
      len = 100;
   }

   Copied (Copied c) : num(c.num), len(c.len) { }  // A

   Copied ( ) : num(NULL), len(0) { }

   virtual Copied operator=(const Copied& c) {
      return Copied(c);
   }
};

int main( ) {
   Copied n1(5);
   Copied n2(n1);
   n2.num[0] = 50;
   cout << "n2.num[0] = " << n2.num[0] << ", n1.num[0] = " << n1.num[0] << endl;
   return 0;
}
```

Question 13 (C++).
The line A above gets a compiler error saying `error: invalid constructor; you probably meant Copied (const Copied&)`. What is the best reason for this error being given by the compiler?

a. All arguments to all functions must be declared `const`

b. All arguments to constructors must be declared `const`

c. To call the constructor of line "A" requires making a copy of the object passed to "c" which would require calling `Copied(Copied c) ...` again, leading to an infinite loop.

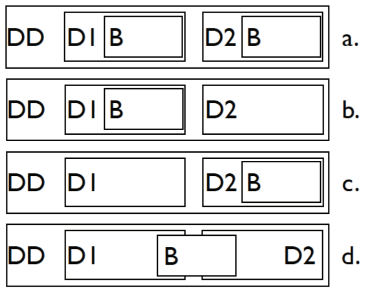
d. The Java compiler was used to compile this C++ program.

e. Initializer lists can only be used on scalars, not arrays or pointers such as `num`.

```
#include <string>
#include <iostream>
using namespace std;
class B {
public:
   B( ) {
      a = 1;
      std::cout << "B::B()" << endl;
   }
   virtual void print( ) {
      cout << "B::a = " << a << endl;
   }
   ~B() {cout << "destroy B" << endl;};
   int a;
};

class D1 : public B {
public:
   D1( ) : B( ) {
      a = 1;
      cout << "D1::D1()" << endl;
   }
   virtual void print( ) {
      cout << "D1::a = " << a << endl;
   }
   ~D1() {cout << "destroy D1" << endl;};
   int a;
};
```

```
class D2 : public B {
public:
   D2( ) : B( ) {
      a = 1; cout << "D2::D2()" << endl;
   }
   virtual void print( ) {
      cout << "D2::a = " << a << endl;
   }
   ~D2() {cout << "destroy D2" << endl;}
   int a;
};

class DD : public D1, public D2 {
public:
   DD( ) : D2( ), D1( ) {
      cout << "DD::DD()" << endl;
   }
   ~DD() {cout << "destroy DD" << endl;};
};

int main(int argc, char * argv[]) {
   DD* d = new DD( );
}
~
```

Question 14 (C++).
Which object most closely resembles what is constructed by the `DD( )` constructor call?

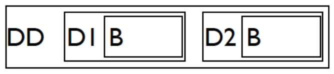


```
#include <string>
#include <iostream>
using namespace std;
class B {
public:
   B( ) {
      a = 1;
      std::cout << "B::B()" << endl;
   }
   virtual void print( ) {
      cout << "B::a = " << a << endl;
   }
   ~B() {cout << "destroy B" << endl;}
   int a;
};

class D1 : virtual public B {
public:
   D1( ) : B( ) {
      a = 1;
      cout << "D1::D1()" << endl;
   }
   ~D1() {cout << "destroy D1" << endl;};
   int a;
};
```

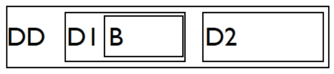
```
class D2 : virtual public B {
public:
   D2( ) : B( ) {
      a = 1;
      cout << "D2::D2()" << endl;
   }
   ~D2() {
      cout << "destroy D2" << endl;
   }
   int a;
};

class DD : public D1, public D2 {
public:
   DD( ) : D1( ), D2( ), B( ) {
      cout << "DD::DD()" << endl;
   }
   ~DD() {cout << "destroy DD" << endl;};
};

int main(int argc, char * argv[]) {
   DD* d = new DD( );
}
```

Question 15 (C++).

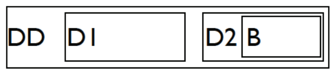
Which object most closely resembles what is constructed by the `DD( )` constructor call?



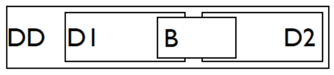
a.



b.



c.



d.

```
#include <string>
#include <iostream>
using namespace std;
class B {
public:
   B( ) {
      a = 1;
      std::cout << "B::B()" << endl;
   }
   virtual void print( ) {
      cout << "B::a = " << a << endl;
   }
   ~B() {cout << "destroy B" << endl;}
   int a;
};

class D1 : virtual public B {
public:
   D1( ) : B( ) {
      a = 1;
      cout << "D1::D1()" << endl;
   }
   ~D1() {cout << "destroy D1" << endl;};
   int a;
};
```

```
class D2 : virtual public B {
public:
   D2( ) : B( ) {
      a = 1;
      cout << "D2::D2()" << endl;
   }
   ~D2() {
      cout << "destroy D2" << endl;
   }
   int a;
};

class DD : public D1, public D2 {
public:
   DD( ) : D1( ), D2( ), B( ) {
      cout << "DD::DD()" << endl;
   }
   ~DD() {cout << "destroy DD" << endl;};
};

int main(int argc, char * argv[]) {
   DD* d = new DD( );
}
```

Question 16 (C++).
In the program above (which is the same as the program for Question 15), whose constructor calls the constructor for the B object(s)?

a. DD's

b. D1's

c. D2's

d. D2 first constructs it, and then D1.

e. D1 first constructs it, and then D2.

```
#include <iostream>
#include <string>
using namespace std;
template<class T> class LLNode {
   LLNode * next;
   T data;
public:
   LLNode(T newData, LLNode* nextNode);
   virtual ~LLNode();
   void print( ) const;
};

template<class T> LLNode<T>::LLNode(
      T newData, LLNode* nextNode) {
   data = newData;
   next = nextNode;
}
```

```
template<class T> void LLNode<T>::print( ) const {
   cout << data << endl;
   if (next != NULL) {
      next->print( );
   }
}
template<class T> LLNode<T>::~LLNode() {
   if (next != NULL) delete next;
}

int main(void) {
   LLNode<int>* head = new LLNode<int>(0, NULL); // A
   for (int i=1; i < 4; i++) {
      head = new LLNode<int>(i, head);
   }
   head->print( );
}
```

Question 17 (C++).
What is the type of the `data` field in the LLNode object pointed to by `head` and created by the `new` operation and constructor call in statement A?

a. `T`

a. `T*` (a pointer to a T)

b. `int`

b. `int*` (a pointer to an int)

```
import java.util.Map;
import java.util.TreeMap;

public class HashMap {
   public static void main(String[] args) {
      // TODO code application logic here
      Map<X, Y> Z = new TreeMap<X, Y>();  //(A)

      // <for brevity code that uses the Map is not shown.
   }
}
```

Question 18 (Java).
In the code above, we wish to define a TreeMap Map that uses `String`s for keys and `Integer`s for the data. The name of the `Map` should be `histogram`. To perform this declaration, `X` in the above code should be

a. String

b. Integer

c. histogram

e. None of the above.

Question 19 (Java).
In the code above, we wish to define a TreeMap Map that uses `String`s for keys and `Integer`s for the data. The name of the `Map` should be `histogram`. To perform this declaration, `Y` in the above code should be

a. String

b. Integer

c. histogram

e. None of the above.

Question 20 (Java).
In the code above, we wish to define a TreeMap Map that uses `String`s for keys and `Integer`s for the data. The name of the `Map` should be `histogram`. To perform this declaration, `Z` in the above code should be

a. String

b. Integer

c. histogram

e. None of the above.

```
public class EB extends Exception {
   public String msg;
   public EB(String s) {msg=s;}
   public String toString( ) {return msg;}
}

public class ED extends EB {
   public String toString( ) {return "from ED"+msg;}
   public ED(String s) {super(s);}
}

public class Thrower {

   public static void heave(int i) throws ED, EB {
      if (i == 0) {throw new EB("i=0");}
      if (i == 1) {throw new ED("i=1");}
      if (i == 2) {throw new ED("i=2");}
   }
}

public class Test {
   static int count = 0;

   public static void main(String[] args) throws Exception {
      try {
         Thrower.heave(2);
         Thrower.heave(1);
         Thrower.heave(0);
      } catch (EB e) {System.out.print("EB: "+e); count++;
      } catch (ED e) {System.out.print("ED: "+e); count++;
      } finally {System.out.println("count: "+count);
      }
   }
}
```

Compiling this code gets the error:

```
javac Test.java
Test.java:16: exception ED has already been caught
      } catch (ED e) {System.out.print("EB: "+e); count++;
        ^
1 error
```

Question 21 (Java).
Pick the best reason why Java gives this error message and put the corresponding letter on the answer sheet.

a. `heave` throws an `ED` exception in two places and only one exception can be caught.

b. The `catch (EB e)` clause will catch the `ED` exception because `ED` is derived from `EB`.

c. There can be only one `catch` for a `try` clause.

d. An exception was thrown and caught while the program was compiled by `javac`.

```
public class EB extends Exception {
   public String msg;
   public EB(String s) {msg=s;}
   public String toString( ) {return msg;}
}

public class ED extends EB {
   public String toString( ) {return " from ED, "+msg;}
   public ED(String s) {super(s);}
}

public class Test {
   static int count = 0;

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

   public void catcher( ) throws EB, ED {
      try {
         heave(1);
         heave(0);
      } catch (ED e) {throw new ED("i=99");
      } catch (EB e) {throw new EB("i=57");
      } finally {System.out.println("finally executed");}
   }
}

class Main {
   public static void main(String[] args) throws Exception {
      Test t = new Test( );
      try {
         t.catcher( );
      } catch (ED e) {System.out.print("ED: "+e);}
      } catch (EB e) {System.out.print("EB: "+e);}
   }
}
```

Question 22 (Java).
What is printed by the above program?

a. `finally executed`
   `EB:  from ED, i=99`

b. `EB:  from ED, i=99`

c. `ED:  from EB, i=57`

d. `EB:  from ED, i=57`

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

class X {
public:
   X( ) { }
   ~X( ) { }
   virtual void foo(int i, double d) {cout << "A" << endl;} // A
   virtual void foo(double d, int i) {cout << "B" << endl;} // B
};

class Y : public X {
public:
   Y( ) { }
   ~Y( ) { }
   virtual void foo(double d, int i) {cout << "C" << endl;} // C
   virtual void foo(int i, int j) {cout << "D" << endl;} // D
   virtual void foo(double d, double f) {cout << "E" << endl; } // E
};

int main()
{
   int i = 0;
   double d = 1.0;
   X* x = new X( );
   Y* y = new Y( );
   x->foo(i,d); // F
   x->foo(d,i); // G
   x->foo(i,i); // H
   x->foo(d,d); // I
   y->foo(i,d); // J
   y->foo(d,i); // K
   y->foo(i,i); // L
   y->foo(d,d); // M
}
```

Question 23 (C++). What is printed by line F? Answer “Err” if the call is ambiguous.
Question 24 (C++). What is printed by line G? Answer “Err” if the call is ambiguous.
Question 25 (C++). What is printed by line H? Answer “Err” if the call is ambiguous.
Question 26 (C++). What is printed by line I? Answer “Err” if the call is ambiguous.
Question 27 (C++). What is printed by line J? Answer “Err” if the call is ambiguous.
Question 28 (C++). What is printed by line K? Answer “Err” if the call is ambiguous.
Question 29 (C++). What is printed by line L? Answer “Err” if the call is ambiguous.
Question 30 (C++). What is printed by line M? Answer “Err” if the call is ambiguous.

```
class X {
   public X( ) { }
   public void foo(int i, double d) {System.out.println("A");} // A
   public void foo(double d, int i) {System.out.println("B");} // B
};

class Y extends X {
   public Y( ) { }
   public void foo(double d, int i) {System.out.println("C");} // C
   public void foo(int i, int j) {System.out.println("D");} // D
   public void foo(double d, double f) {System.out.println("E");} // E
};

class Main {
   public static void main(String[] args) {
      int i = 0;
      double d = 1.0;
      X x = new X( );
      Y y = new Y( );
      x.foo(i,d); // F
      x.foo(d,i); // G
      x.foo(i,i); // H
      x.foo(d,d); // I
      y.foo(i,d); // J
      y.foo(d,i); // K
      y.foo(i,i); // L
      y.foo(d,d); // M
   }
}
```

Question 31 (Java). What is printed by line F? Answer "Err" if the call is ambiguous.
Question 32 (Java). What is printed by line G? Answer "Err" if the call is ambiguous.
Question 33 (Java). What is printed by line H? Answer "Err" if the call is ambiguous.
Question 34 (Java). What is printed by line I? Answer "Err" if the call is ambiguous.
Question 35 (Java). What is printed by line J? Answer "Err" if the call is ambiguous.
Question 36 (Java). What is printed by line K? Answer "Err" if the call is ambiguous.
Question 37 (Java). What is printed by line L? Answer "Err" if the call is ambiguous.
Question 38 (Java). What is printed by line M? Answer "Err" if the call is ambiguous.

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

class B {

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

   virtual void m1(double ff) {
      cout << "A" << endl;
   }

   virtual void m1(int i ) {
      cout << "B" << endl;
   }

   virtual void m2(int i ) {
      cout << "C" << endl;
   }
};

class D1 : public B {
public:
   D1( ) { }

   ~D1( ) { }

   virtual void m2(int i) {
      cout << "D" << endl;
   }
};

class D2 : public D1 {
public:
   D2( ) { }

   ~D2( ) { }

   virtual void m1(int i) {
      cout << "E" << endl;
   }
};
```

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

   D1* d1P = new D1( );
   D2* d2P = new D2( );
   B* bP = new B( );
   B bo = *d2P;
   B& br = *d1P;

   d2P->m1(1);   // A
   d2P->m1(1.0); // B
   bP->m1(1);   // C
   bP->m1(1.0); // D
   bP->m2(1.0); // E
   br.m1(1);    // F
   br.m1(1.0);  // G
   bo.m1(1);     // H
   bo.m1(1.0);   // I

   bP = d2P;

   bP->m1(1);   // J
   bP->m1(1.0); // K
   bP->m2(1); // L

   bP = new D1( );

   bP->m1(1);   // M
   bP->m1(1.0); // N
   bP->m2(1); // O
}
```

Question 39 (C++). What is printed by line A?
Question 40 (C++). What is printed by line B?
Question 41 (C++). What is printed by line C?
Question 42 (C++). What is printed by line D?
Question 43 (C++). What is printed by line E?
Question 44 (C++). What is printed by line F?
Question 45 (C++). What is printed by line G?
Question 46 (C++). What is printed by line H?

Question 47 (C++). What is printed by line I?
Question 48 (C++). What is printed by line J?
Question 49 (C++). What is printed by line K?
Question 50 (C++). What is printed by line L?
Question 51 (C++). What is printed by line M?
Question 52 (C++). What is printed by line N?
Question 53 (C++). What is printed by line O?