



READ THIS NOW!

- Print your name in the space provided below.
- Print your name and ID number on the Opscan form; be sure to code your ID number on the Opscan form. Code **Form A** on the Opscan.
- Choose the single best answer for each question — some answers may be partially correct. If you mark more than one answer, it will be counted wrong.
- Unless a question involves determining whether given C++ code is syntactically correct, assume that it is valid. The given code has been compiled and tested, except where there are deliberate errors. Unless a question specifically deals with compiler `#include` directives, you should assume the necessary header files have been included.
- Be careful to distinguish integer values from floating point (real) values (containing a decimal point). In questions/answers which require a distinction between integer and real values, integers will be represented without a decimal point, whereas real values will have a decimal point, [2704 (integer), 2704.0 (real)].
- The answers you mark on the Opscan form will be considered your official answers.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- This is a closed-book, closed-notes examination. No calculators or other electronic devices may be used during this examination. You may not discuss (in any form: written, verbal or electronic) the content of this examination with any student who has not taken it. You must return this test form when you complete the examination. Failure to adhere to any of these restrictions is an Honor Code violation.
- There are 25 questions, equally weighted. The maximum score on this test is 100 points.

Do not start the test until instructed to do so!

Print Name (Last, First) _____

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

signature

OOP Identification

Given the description below of a Compact Disc (CD) mail order company:

A request for one or more CDs is received by the order department from a customer, (uniquely identified by their account number). The availability of the quantity of each CD ordered is checked in the stock records. For the CDs that are not in stock a back order is placed with the purchase department and a backorder notice is mailed to the customer. For the CDs of the order that are in stock, a bill is prepared and sent to the shipping department. A shipping clerk packages the CDs, updates the stock records, and mails the CDs to the customer. Customers may cancel backordered CDs by returning a cancellation post card. If a cancellation card is received before the backordered stock is received then the customer's request is amended, but the purchase department does not delete the backorder.

Use the following responses for the next 6 questions.

- (1) object (2) class (3) attribute (4) service
(5) actor (6) none of the above

Identify each of the following using the above CD mail order description and responses:

- #1 Account Number
- #2 Customer
- #3 Order
- #4 Package
- #5 Shipping Clerk
- #6 CD Availability

#7 Separation of the class interface from class implementation supports the software engineering goal of flexibility because

- (1) It allows encapsulation of objects, and requires functions to access the objects only through the interface functions.
- (2) It allows class implementations to be swapped without affecting other parts of the program.
- (3) It allows creation of many different objects as class instances, instead of just one object.
- (4) It allows us to write many different classes

#8 We consider scenarios in design because

- (1) Large systems can be complicated and difficult to understand completely
- (2) Scenarios allow us to understand the responsibilities of the objects in a system
- (3) Scenarios allow us to view a single aspect of a system without trying to understand the whole system at one time.
- (4) All of the above.

#9 In identifying responsibilities of objects in a system, if the system description includes a statement of the form “A player can move any of their game pieces to an adjacent square on the board”, which of the following assignments of responsibility matches best with this statement? Assume four classes `Player`, `GamePiece`, `Square` and `Board`.

- (1) The `Player` class should have a function with a name like `movePiece`
- (2) The `GamePiece` class should have a function with a name like `move`
- (3) The `Square` class should have a function with a name like `placePiece`
- (4) The `Board` should have a function with a name like `movePiece`

#10 A class/object should be eliminated from a design for a system because

- (1) The functions of the class would all be really short
- (2) The class/object has no role in the system
- (3) Both a and b
- (4) None of these

#11 Suppose you have identified several responsibilities for a class, but they don't seem to go together. However, you can divide the responsibilities into two classes. Which of the following is the most appropriate action?

- (1) Eliminate the class
- (2) Divide the class into two classes
- (3) Replace the class with a more general class
- (4) Eliminate some of the responsibilities

#12 Suppose you have identified several responsibilities for a class, but for a few of them you can't come up with a scenario in which they would be necessary. Which of the following is the most appropriate action?

- (1) Eliminate the class
- (2) Divide the class into two classes
- (3) Replace the class with a more general class
- (4) Eliminate some of the responsibilities

Objects and Memory

For the following 3 questions, consider the class `Aggregate` as declared below. Assume that the class `AggNode` has one `int` field.

```
class Aggregate {
public:
    Aggregate() : fst(new AggNode(0)) {}
    Aggregate(const Aggregate& a)
        : fst(new AggNode(*(a.fst))) {}
    Aggregate& operator= (const Aggregate& a) {
        if (this != &a) {
            delete fst;
            fst = new AggNode(*(a.fst));
        }
        return *this;
    }
    void reset(const int& val) {
        delete fst;
        fst = new AggNode(val);
    }
    ~Aggregate() { delete fst; }
private:
    AggNode* fst;
};
```

#13 What memory problem would occur if this class is used as follows

<pre>void main () { Aggregate a1; a1 = f(); }</pre>	<pre>Aggregate f() { Aggregate a; return a; }</pre>
---	---

- (1) Dangling pointer
- (2) Memory leak
- (3) Alias (but no abnormal termination)
- (4) None of these.

#14 What memory problem would occur if we use the same functions as in question #13, but change the copy constructor so that it has the form

```
Aggregate(const Aggregate& a) : fst(a.fst) {}
```

- (1) Dangling pointer
- (2) Memory leak
- (3) Alias (but no abnormal termination)
- (4) None of these.

#15 Again, what memory problem would occur if we use the same functions as in question #13, but allowed the compiler to create a copy constructor for us?

- (1) Dangling pointer
- (2) Memory leak
- (3) Alias (but no abnormal termination)
- (4) None of these.

For the following 4 questions consider the following declaration of the Association and Associate classes.

```
class Associate {
public:
    Associate() : x(0) {}
    void doSomething() { cout << "Wow, x is " << x << endl; }
private:
    int x;
};

class Association {
public:
    Association() : aptr(NULL) {}
    Association(const Association& a) : aptr(NULL) {}
    void add(Associate& a) { aptr = &a; }
    void use() { aptr->doSomething(); }
    Association& operator= (const Association& a) { aptr = NULL; }
    ~Association() {}
private:
    Associate* aptr;
};
```

#16 What memory problems would occur if these classes were used as follows?

```
void main() {
    Association a;
    Associate b;
    a.add(b);
    a.use();
}
```

- (1) Dangling pointer
- (2) Memory leak
- (3) Alias (but no abnormal termination)
- (4) None of these.

#17 Suppose we changed the destructor of the Association class to read

```
~Association() { delete aptr; }
```

and used the main() function in question #16, what output would be seen?

- (1) Wow, x is 0
- (2) Wow, x is 1
- (3) No output would be seen, because the program would not compile.
- (4) No output would be seen, because a runtime error would occur first.

#18 Suppose we changed the destructor of the Association class to read

```
~Association() { delete aptr; }
```

and used the functions f and main() below, what output would be seen?

<pre>void main() { Association a; f(a); a.use(); }</pre>	<pre>void f(Association a) { Associate b; a.add(b); }</pre>
--	---

- (1) Wow, x is 0
- (2) Wow, x is 1
- (3) No output would be seen, because the program would not compile.
- (4) No output would be seen, because a runtime error would occur first.

#19 If we changed the copy constructor of the `Association` class so that it does a shallow copy, it would look like:

- (1) `Association (const Association& a) : aptr(NULL) {}`
- (2) `Association (const Association& a) : aptr(a.aptr) {}`
- (3) `Association (const Association& a) : aptr(new Associate()) {}`
- (4) `Association () : aptr(NULL) {}`

Class Basics

Consider the following class declaration and implementation:

```
class Dimension {
private:
    int width, height;
public:
    Dimension(): width(0), height(0){};
    Dimension(int w, int h): width(w), height(h){};
    Dimension(const Dimension& d): width(d.width), height(d.height){};
    int getWidth() const {return width;};
    int getHeight() const {return height;};
    void setWidth (int w) {width = w;};
    void setHeight(int h) {height = h;};
};
```

#20 Given the array declaration below:

```
Dimension Compass[4];
```

what are the values of the data members, width and height, of `Compass[3]`?

- | | | |
|--------------------------|-----------------------|----------|
| (1) undefined, undefined | (2) 0, 0 | (3) w, h |
| (4) INT_MIN, INT_MIN | (5) INT_MAX, INT_MAX | (6) 3, 3 |
| (7) 4, 4 | (8) none of the above | |

#21 True/False:

In order to operate correctly, the above `Dimension` class should have an over-loaded assignment operator implementation added to the class?

- (1) True (2) False

Class Basics (continued)

Consider the class declaration and implementation:

```
class RanList {
private:
    int Num, Lim;
    int* Ray;
public:
    RanList(int N=1, int L=INT_MAX);
    int Avg();
    ~ RanList();
};

int RanList::Avg() {

    int tSum = 0;
    for (int i = 0; i < Num; i++)
        tSum += Ray[i];
    return (tSum/Num);
}
```

```
RanList::RanList (int N, int L) {
    Num = N;
    Lim = L;
    Ray = new int[Num];
    for (int i = 0; i < Num; i++)
        Ray[i] = rand() % Lim;
}

RanList::~RanList () {
    delete [] Ray;
}
```

#22 Consider the code fragments below. In which fragment does a memory leak occur?

<pre>// (1) int sumavg = 0; for (int i=0; i<30; i++) { for (int j=0; j <1; j++) { RanList r(30,30); sumavg += r.Avg(); } } cout <<"Avg of Avgs="<<sumavg/30<<endl;</pre>	<pre>// (2) RanList* r; int sumavg = 0; for (int x=0; x<30; x++) { r = new RanList(30,30); sumavg += r->Avg(); } cout <<"Avg of Avgs="<<sumavg/30<<endl;</pre>
<p>(3) Both 1 and 2 contain memory leaks</p>	<p>(4) None of the above contains memory leaks.</p>

#23 If class RanList contained a copy constructor in which of the following would it **NOT** be automatically called?

- (1) When in the same statement a RanList object is defined and initialized to an existing RanList object (e.g., RanList s(r);)
- (2) When a RanList object is passed to a function as a value parameter.
- (3) When a RanList object is assigned to another RanList object.
- (4) When a RanList object is returned by a function.
- (5) both 1 and 3 (6) both 2 and 4 (7) none of the above

Consider the classes PerLib and Book:

```
class PerLib {
private:
    string Owner;
    int    numBooks;
    Book*  collection;
public:
    PerLib(string O, int numB);
    bool AddBook(const Book& B);
    Book getBook(string title) const;
    ~ PerLib();
};

PerLib::PerLib(string O, int numB) {
    Owner = O;
    numBooks = numB;
    collection = new Book[numB];
}

. . .
PerLib::~ PerLib () {
    delete []collection;
}
}
```

```
class Book {
private:
    string Title, Author;
    int    Pages;
public:
    Book( string T= "",
          string A= "",
          int P = 0 );
    string getTitle() const;
    string getAuthor() const;
    int    getPages() const;
};
```

Assume a function is implemented to compute the total number of pages in a PerLib (Personal Library):

```
int Pages(PerLib PL) {
    // Correct code to compute and return the total number
    // of all the pages in all books.
}
```

#24 Calling the function Pages () will have an unfortunate side effect (even though the body of the function is correct). Which of the following terms best labels the side effect?

- (1) Memory Leak (2) Dangling Pointer
- (3) Mutable Pointer (4) None of the above

#25 Which of the following changes can be implemented to correct the side effect?

- (1) Add a copy constructor to PerLib (2) Pass PL by const reference
- (3) Make Pages a PerLib member function (4) Make Pages a const function
- (5) Both (1) and (2) (6) Both (3) and (4)
- (7) None of the above