

Lesson 3 - Page 47 - at end of paragraph of text under "Analysis" heading:

"carries sign information in the former."

should be

"carries sign information in the latter."

Lesson 7 - Page 163 - Listing 7.7

```
28:      cout << "Area of cylinder is: " << Area (radius) << endl;
```

should be

```
28:      cout << "Area of circle is: " << Area (radius) << endl;
```

Lesson 7 - Page 174 - at end of first sentence at top of page:

"effectively sorting the collection in an ascending order."

should be

"effectively sorting the collection in a descending order."

Lesson 9 - Page 226 - last paragraph on page:

"Note how the constructor initializes integer age to zero. Should you forget to SetAge() on a newly constructed object, you can rest assured that the constructor would have ensured that the value contained in variable age is not a random integer (that might look valid) but instead a zero."

should be:

"Note how the constructor initializes integer age to 1. Should you forget to SetAge() on a newly constructed object, you can rest assured that the constructor would have ensured that the value contained in variable age is not a random integer (that might look valid) but instead a 1."

Lesson 9 - Page 230 - last line of code on page:

```
"Human eve("Eve", 18); // eve.age is assigned 18 as specified"
```

should be:

```
"Human eve("Eve", 18); // eve.age is assigned 18 as specified"
```

Lesson 9 - Page 236 - first sentence in paragraph under "Analysis" heading:

"This class basically encapsulates a C-style string in MyString::buffer and relieves you of the task of allocating memory; it deallocates the same every time you need to use a string."

should be:

"This class basically encapsulates a C-style string in MyString::buffer and relieves you of the task of allocating memory; it allocates the same every time you need to use a string."

Lesson 11 - Page 315 - line of code in middle of page:

```
// Func2 overrides Base::Func2()
```

should be:

```
// Func1 overrides Base::Func1()
```

Lesson 11 - Page 316 - line of code near middle of page:

```
CDerived objDerived;
```

should be:

```
Derived objDerived;
```

Lesson 11 - Page 323 - text in analysis following Listing 11.7:

```
duckBilledP.Mammal::Animal::age = 25;  
duckBilledP.Bird::Animal::age = 25;  
duckBilledP.Reptile::Animal::age = 25;
```

should be:

```
duckBilledP.Mammal::age = 25;  
duckBilledP.Bird::age = 25;  
duckBilledP.Reptile::age = 25
```

Lesson 11 - Page 331 - text in analysis following Listing 11.9

“It also features a virtual destructor for class Fish in Line 8. Lines 52–56 in main() demonstrate how a static array of pointers to base class Fish* has been declared and individual elements assigned to newly created objects of type Tuna, Carp, Tuna, and Carp, respectively.”

should be:

“It also features a virtual destructor for class Fish in Line 8. Lines 52–56 in main() demonstrate how a static array of pointers to base class Fish* has been declared and individual elements assigned to newly created objects of type Tuna, Carp, BluefinTuna, and Carp, respectively.”

Lesson 12 - Page 355 - Listing 12.7

Code in Lines 12 – 23 should be:

```
12:     bool operator< (const Date& compareTo)  
13:     {  
14:         if (year < compareTo.year)  
15:             return true;  
16:         else if ((year == compareTo.year) && (month < compareTo.month))  
17:             return true;  
18:         else if ((year == compareTo.year) && (month == compareTo.month)  
19:                 && (day < compareTo.day))  
20:             return true;  
21:         else  
22:             return false;  
23:     }
```

Appendix E - Page 731 – Answers for Lesson 11, Exercises

Code in Exercise 1 should be:

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

class Shape
{
public:
    virtual double Area() = 0;
    virtual void Print() = 0;
};

class Circle: public Shape
{
    double Radius;
public:
    Circle(double inputRadius) : Radius(inputRadius) {}

    double Area() override
    {
        return 3.1415 * Radius * Radius;
    }

    void Print() override
    {
        cout << "Circle says hello!" << endl;
    }
};

class Triangle: public Shape
{
    double Base, Height;
public:
    Triangle(double inputBase, double inputHeight) : Base(inputBase),
    Height(inputHeight) {}

    double Area() override
    {
        return 0.5 * Base * Height;
    }

    void Print() override
    {
```

```
        cout << "Triangle says hello!" << endl;
    }
};

int main()
{
    Circle myRing(5);
    Triangle myWarningTriangle(6.6, 2);

    cout << "Area of circle: " << myRing.Area() << endl;
    cout << "Area of triangle: " << myWarningTriangle.Area() << endl;

    myRing.Print();
    myWarningTriangle.Print();

    return 0;
}
```