

Chapter 1:

15. Consider the class declaration

```
class demoClass
{
    public:
        // assign arguments as initial values for the data members
        demoClass(int a = 5, int b = 10);
        // function returns the maximum of itemA and itemB
        // int max() const;
    private:
        int itemA, itemB;
};
```

- (a) Give the implementation of the constructor using an initialization list.
- (b) Implement the member function max() as an external function.
- (c) For each of the following object declarations, give the corresponding initial values for attributes itemA and itemB:

```
demoClass obj1(7,9); // itemA = _____ itemB = _____
demoClass obj2(12); // itemA = _____ itemB = _____
demoClass obj3;     // itemA = _____ itemB = _____
```

- (d) What is the output from each statement, assuming the previous declarations?

```
cout << obj2.max(); // Output: _____
cout << obj3.max(); // Output: _____
```

16. Identify the syntax errors in the C++ class declarations. There could be several in each class.

<p>(a) className: class</p> <pre>public { className(a, b): void setData(int a, b); } private { int itemA, itemB };</pre>	<p>(b) class className</p> <pre>{ public@ void className(int initValue) getValue() : int; private; int value; }</pre>
---------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------

17. The *accumulator* class describes objects that maintain running totals. In the class, the constructor has a default argument of 0.0, and the `addValue()` member function adds a value to the total. If its argument is omitted, the default value of 1.0 is added. The following is the declaration for the accumulator class:

```
class accumulator
{
    public:
        accumulator(double value = 0.0);    // constructor
        double getTotal() const;           // return total
        void addValue(double value = 1.0); // add value to total
    private:
        double total;
        // total accumulated by the object
};
```

- (a) Give an implementation of the constructor that uses an initialization list.
(b) Implement the member functions `addValue()` and `getTotal()` as external functions.
(c) Trace the program, and determine the result of each output statement. Assume the accumulator class is declared and implemented in the header file "accum.h".

```
#include <iostream>

#include "accum.h"

using namespace std;

int main()
{
    accumulator obj;

    cout << obj.getTotal() << endl;    // output: _____
    obj.addValue(3);
    obj.addValue(obj.getTotal()+3);
    cout << obj.getTotal() << endl;    // output: _____

    // using the constructor, create an object and
    // assign it to obj
    obj = accumulator(8);
    obj.addValue();
    cout << obj.getTotal() << endl;    // output: _____

    return 0;
}
```

28. Trace the program by using the two given versions of the function func(). In each case, give the output provided by the "cout" statements.

```
int main()
{
    int a = 3, b = 8, c = 5, d = 3;

    func(a, b);
    cout << a << " " << b << endl; // Output: _____

    func(c, d);
    cout << c << " " << d << endl; // Output: _____

    return 0;
}
```

- (a) Use version #1 for func(). (b) Use version #2 for func().

```
void func(int x, int& y)          void func(int& x, int& y)
{                                  {
    if (x < y)                    if (x < y)
        x = 2 * y;                x = 2 * y;
    else                            else
        y = 3 * x;                y = 3 * x;
}
```