

## Shorter Answers

*Blanks may stand for one or more words.*

1. Copying or assigning a derived class object to a base class  $\ell$ -value is called \_\_\_\_\_.
2. An overloaded operator must have at least one \_\_\_\_\_ that is \_\_\_\_\_.
3. To be overloaded by a member function, An operator must be a member of the class of its \_\_\_\_\_.
4. The number of \_\_\_\_\_ (or arguments) of an operator is called its \_\_\_\_\_.
5. The fact that  $3+5*x$  parses as  $3+(5*x)$  results from the relative \_\_\_\_\_ of the operators “+” and “\*”.

## Short Answers

6. Text books claim that static member functions cannot access non-static member variables. Explain more clearly what this means in terms of this pointers and memory diagrams.
7. Can non-static member functions access static member variables?
8. Can derived class non-static member functions access private static member variables of a base class?

9. Can dead Snakes eat live Lizards?
10. Give three restrictions that apply to operator overloading in C++.
11. Which of the following pieces of code is most likely correct?
- (a) `Fred* Gertrude::operator->(Fred& f) { return &this->f ; }`
  - (b) `Fred* Gertrude::operator->() { return this ; }`
  - (c) `Fred& Gertrude::operator->() { return &this ; }`
  - (d) `Fred* Gertrude::operator->() { return &*&this ; }`
12. Which of the following pieces of code is most likely correct?
- (a) `Terble operator++(Terble& t, int z)`  
`{ Terble s(t) ; ++t ; return s ; }`
  - (b) `Terble& Terble::operator++(int z)`  
`{ Terble s(*this) ; ++*this ; return s ; }`
  - (c) `Terble Terble::operator++(int z)`  
`{ Terble s(*this) ; ++s ; return *this ; }`
  - (d) `Terble& operator++(Terble& t, int z)`  
`{ return t+=z ; }`
13. List three operators that are likely to return by reference.
14. List three operators that are unlikely to return by reference.
15. What operator almost always acts on two non-constant  $\ell$ -values?
16. Which of the following operators *cannot* be overloaded?
- `[]`
  - `? :`
  - `()`
  - `->`
  - `++`
  - `=`
  - `::`
  - `.`
  - `sizeof`
17. Which of the following operators can be overloaded, but must be overloaded by a member function?

- |       |      |          |
|-------|------|----------|
| • []  | • -> | • ::     |
| • ? : | • ++ | • .      |
| • ( ) | • =  | • sizeof |

18. Is it possible to create an overloaded ternary operator? Explain.
19. Given the declaration "char box[6];" what values do the following expressions yield?
- sizeof(box)
  - sizeof(box[0])
  - sizeof("box[0]")
  - sizeof(&box[0])
20. What happens if you override a public virtual member function of a base class with a private member function in a derived class?

## Longer Answers

21. Explain when and why destructors should be declared virtual. Draw a memory diagram of what might go wrong when someone mistakenly forgets to declare a virtual destructor.
22. What, if anything, is wrong with overloading operator<< as follows?

```
std::ostream& operator<<(std::ostream& out, Employee e)
{
    e.printOn(out) ;
    return out;
}
```

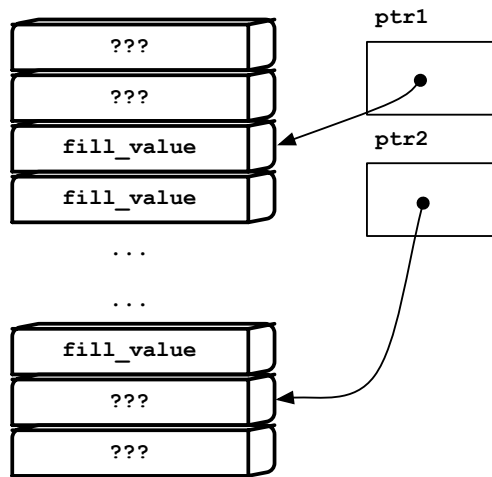


Figure 1: "After" picture for fill (Question 24).

## Coding

23. Given two pointers to doubles, `ptr1` and `ptr2`, write an expression that generates a pointer that points to approximately half-way between `ptr1` and `ptr2`.
24. Write a function `fill` that takes three arguments:
- a pointer to double `ptr1`,
  - a pointer to double `ptr2`,
  - a double value `fill_value`,
- that fills in the memory between `ptr1` (inclusive) and `ptr2` (exclusive) with the value `fill_value`. (See Figure 1.)
25. Give an example of how to use a colon-list to initialize member variables.
26. Show how to declare a `Professor`-class that is derived from both `Employee` and `Tenurable`.
27. Given an `Employee` class with declaration

```
class Employee
{
    public:
```

```
virtual ~Employee() ;  
virtual void printOn(std::ostream& out) const ;  
protected:  
Employee() ;  
void setEmployeeNumber() ;  
private:  
int myEmployeeNumber ;  
} ;// end class Employee
```

show how to overload the "<<" operator to print an Employee object. Will this operator likely work objects from a derived class?

## True and False

Circle **TRUE** or **FALSE** as appropriate. Questions that don't clearly indicate *one* choice shall be marked wrong.

1. Operators overloaded by member functions can be virtual.  
**TRUE FALSE**
2. Operators overloaded by non-member functions can be virtual.  
**TRUE FALSE**
3. Overloading the + operator with several different signatures results in *run-time polymorphism*.  
**TRUE FALSE**
4. A class with one *pure virtual* member function and one non-pure member function is an *abstract* class.  
**TRUE FALSE**
5. It is legal to declare multiple constructors for an abstract class.  
**TRUE FALSE**
6. It is possible to determine all functions that can access the private members of a class by reading the class definition.  
**TRUE FALSE**
7. It is possible to determine all functions that can access the protected members of a class by reading the class definition.  
**TRUE FALSE**

## Memory Diagrams

28. Complete the following table:

Time of:	Allocation	Construction	Destruction	Deallocation
Global Variables				
Static function local Variables				
Temporary stack expressions				
Heap objects				

29. This is in part a trick question, so explain your answer carefully.

- How much space do non-static member variables take? Where in a memory diagram would you find them?
- How much space do static member variables take? Where in a memory diagram would you find them?

30. Use a memory diagram to explain what a *dangling pointer* is and why the following code segment is dangerous.

```
double * x_ptr ;
if (true) {double y ; x = & y ;}
*x_ptr = 5.3 ;
```