Quiz 6, IS0020, Feb 17, 2004

Name:

---

1. The correct function name for overloading the addition (+) operator is
   (a) **`operator+`**
   (b) **`+operator`**
   (c) **`operator(+)`**
   (d) **`operator:+`**

ANS: (a)

2. Which statement about operator overloading is false?
   (a) New operators can never be created.
   (b) Certain overloaded operators can change the number of arguments they take.
   (c) The precedence of an operator cannot be changed by overloading.
   (d) Overloading cannot change how an operator works on built-in types.

ANS: (b)

3. To overload the **`+=`** operator
   (a) only the **`+`** operator needs to be overloaded.
   (b) only the **`+`** and **`=`** operators need to be overloaded.
   (c) the **`+=`** operator must be explicitly overloaded.
   (d) the **`+`** and **`=`** operators need to be overloaded implicitly.

ANS: (c)

4. Which situation would require the use of a non-member overloaded operator?
   (a) The overloaded operator is **`=`**.
   (b) The left most operand must be a class object or a reference to a class object of the operator's class.
   (c) The left operand is an **`int`**.
   (d) The operator returns a reference.

ANS: (c)

5. An overloaded **`+`** operator takes a class object and a **`double`** as operands. For it to be commutative (i.e., **`a + b`** and **`b + a`** both work),
   (a) **`operator+`** must be a member function of the class from which the objects are instantiated.
   (b) **`operator+`** must be a non-member function.
   (c) the **`operator+`** function that takes the object as the left operand must be a member function, and the other **`operator+`** must be a non-member function.
   (d) both **`operator+`** functions must be non-member **`friend`** functions of the class.

ANS: (c)

6. Suppose you have a programmer-defined data type **`Data`** and want to overload the **`<<`** operator to output your data type to the screen in the form **`cout << dataToPrint;`** and allow cascaded function calls. The first line of the function definition would be
   (a) **`ostream &operator<<(ostream &output, const Data &dataToPrint)`**
   (b) **`ostream operator<<(ostream &output, const Data &dataToPrint)`**
   (c) **`ostream &operator<<(const Data &dataToPrint, ostream &output)`**

(d) **`ostream operator<<(const Data &dataToPrint, ostream &output)`**

ANS: (a)

7. **`y`** and **`z`** are user-defined objects and the **`+=`** operator is an overloaded member function. The operator is overloaded such that **`y += z`** adds **`z`** and **`y`**, then stores the result in **`y`**. Which of the following expressions is equivalent to **`y += z`**?
   (a) **`y = (y.operator+=) + (z.operator+=)`**
   (b) **`y.operator+=( z )`**
   (c) **`y = y + z`**
   (d) **`y.operator+=( z ) = y.operator+=( z ) + z.operator+=( z )`**

ANS: (b)

8. For non-**`static`** overloaded member functions,
   (a) binary operators can have two arguments and unary operators can have one.
   (b) both binary and unary operators take one argument.
   (c) binary operators can have one argument, and unary operators cannot have any.
   (d) neither binary nor unary operators can have arguments.

ANS: (c)

9. The conventional way to distinguish between the overloaded preincrement and postincrement operators (**`++`**) is
   (a) to assign a dummy value to preincrement.
   (b) to make the argument list of postincrement include an **`int`**
   (c) to have the postincrement operator call the preincrement operator.
   (d) implicitly done by the compiler.

ANS: (b)

10. There exists a data type **`Date`** and member function **`Increment`**. The **`++`** operator is being overloaded to postincrement an object of type **`Date`**. Select the correct implementation.

(a)
```
Date Date::operator++( int ) {
     Date temp = *this;
     Increment();
     return *temp;
}
```

(b)
```
Date Date::operator++( int ) {
     Increment();
     Date temp = *this;
     return temp;
}
```

(c)
```
Date Date::operator++( int ) {
     Date temp = *this;
     return this;
     temp.Increment();
}
```

(d)
```
Date Date::operator++( int ) {
     Date temp = *this;
     Increment();
     return temp;
}
```

ANS: (d)