 Answers to review questions from Chapter 20

1. Suppose that you have a **Set<int>** variable called **primes**. How would you use an iterator to display every element in **primes** in ascending order, without using a range‑based **for** loop?

for (Set<int>::iterator it = primes.begin();

it != primes.end(); it++) {

cout << \*it << endl;

}

2. Assuming that the variable **it** is an iterator, describe the effect of the expression **\*it++**.

**As with pointers, the expression \*it++ refers to the current value of the iterator but has the side effect of advancing the iterator to the next element.**

3. True or false: If **c** is a nonempty collection, calling **c.begin()** returns an iterator pointing at the first element of that collection.

**True.**

4. True or false: If **c** is a nonempty collection, calling **c.end()** returns an iterator pointing at the last element of that collection.

**False. The end method returns an iterator that indicates the nonexistent element after the last one.**

5. In the UML diagram for the iterator hierarchy in Figure 20‑1, the list of methods in the **ForwardIterator** box is empty. Why does this level of the hierarchy exist?

**The ForwardIterator class uses multiple inheritance to combine the characteristics of an InputIterator and an OutputIterator.**

6. True or false: If **c** is a nonempty collection, calling **c.end()** returns an iterator pointing at the last element of that collection.

**False. This question was a duplicate of question 4 left in by accident.**

7. What feature of the von Neumann architecture makes it possible to define pointers to functions?

**All instructions inside a von Neumann machine are stored in the same memory cells as those used for data. Because of this design, every function or method has an address.**

8. Describe the difference between the declarations

char \*f(string);

and

char (\*f)(string);

**The first declaration is the prototype for a function f that takes a C++ string and returns a pointer to a character. The second declares a function variable that can hold the address of any function taking a C++ string and returning a character.**

9. How would you declare a variable **fn** as a pointer to a function taking two integers and returning a Boolean value?

bool (\*fn)(int, int);

10. What is a *callback function?*

**A *callback function* is a function value passed by a client into the implementation of an abstraction. The implementation can then call this function internally to run client‑supplied code.**

11. What is a *mapping function?*

**A *mapping function* is a function that applies a callback function to every element in a collection.**

12. In your own words, describe the difference between a *function pointer* and a *function object.*

**A *function pointer* is the address of a function in memory. A *function object* is an object that overrides the definition of operator(). Both strategies produce a functional entity that can be applied using the traditional parenthesis notation.**

13. By definition, every function class implements one particular operator. Which operator is it?

**operator()**

14. What are the two arguments to the **sort** function exported by the STL **<algorithm>** library?

**The two arguments are both iterators indicating the same collection. The sort function sorts the elements between the two iterators.**

15. What are the primary attributes of the functional programming paradigm, as it is described in this chapter?

**• Programs are expressed in the form of nested function calls that perform the necessary computation without performing any operations (such as assignment) that change the program state.**

**• Functions are data values and can be manipulated by the programmer just like other data values.**

16. True or false: The types of the two arguments in a function class that inherits from **binary\_function** must be the same.

**False.**

17. What is the purpose of including the **bind1st** and **bind2nd** functions in the **<functional>** library?

**These functions make it possible to create a new function that specifies either the first or second argument of an existing binary function.**

18. Use the capabilities of the **<functional>** library to write a single call to **count\_if** that returns the number of even values in the integer vector **vec**.

bool isEven(int n) { return n % 2 == 0; }

count\_if(v.begin(), v.end(), isEven)

19. What is a *comparison function?*

**A *comparison function* takes two arguments and returns a Boolean value that indicates the result of the comparison. The comparison functions listed in Table 20‑2 are equal\_to, not\_equal\_to, less, greater, less\_equal, and greater\_equal.**

20. Describe the type introduced by the following **typedef** statement:

typedef void (\*proc)();

**This statement defines proc to be the type space of functions that take no arguments and return no results.**

21. List the complete set of operators implemented by an iterator offering the level of service provided by **RandomAccessIterator**.

**As shown in Figure 20‑1, the RandomAccessIterator class inherits all the operations from its superclasses. The complete set as given in the figure is operator\*, operator++, operator==, operator!=, operator->, operator+, operator-, operator+=, operator-=, operator<, operator>, operator<=, and operator>=.**

22. True or false: Pointers in C++ define all the operators necessary to implement a **RandomAccessIterator**.

**True.**