 Answers to review questions from Chapter 5

1. True or false: An abstract data type is one defined in terms of its behavior rather than its representation.

**True.**

2. What three advantages does this chapter cite for separating the behavior of a class from its underlying implementation?

**1. Simplicity**

**2. Flexibility**

**3. Security**

3. What is the STL?

**The STL is the Standard Template Library, which defines a standard set of collection classes.**

4. If you want to use the **Vector** class in a program, what **#include** line do you need to add to the beginning of your code?

#include "vector.h"

5. List at least three advantages of the **Vector** class over the more primitive array mechanism available in C++.

**1. Vectors can change their size dynamically.**

**2. Vectors make their size available to the programmer.**

**3. Vectors support the insertion and deletion of elements.**

**4. Vectors ensure that accesses to elements do not fall outside the bounds of the vector.**

6. What is meant by the term *bounds‑checking?*

**Bounds-checking consists of testing the index of a vector to ensure that it stays within the vector boundaries.**

7. What is a *parameterized type?*

**A parameterized type is one that includes the specification of a type for its subsidiary elements. The libraries, for example, support the declaration of a Vector<int> or a Vector<string>.**

8. What type name would you use to store a vector of Boolean values?

**Vector<bool>**

9. True or false: The default constructor for the **Vector** class creates a vector with ten elements, although you can make it longer later.

**False.**

10. How would you initialize a **Vector<int>** with 20 elements, all equal to 0?

new Vector<int>(20, 0)

11. What method do you call to determine the number of elements in a **Vector**?

**The size method.**

12. If a **Vector** object has *N* elements, what is the legal range of values for the first argument to **insert**? What about for the argument to **remove**?

**The insert method allows the first argument to range from 0 (insert at the beginning) up to *N* (insert at the end). The remove method allows index values ranging from 0 (remove the first element) up to *N* – 1 (remove the last element).**

13. What feature of the **Vector** class makes it possible to avoid explicit use of the **get** and **set** methods?

**The Vector class overloads the bracket selection operator to support traditional index notation, as in vec[i].**

14. Why is it important to pass vectors and other collection objects by reference?

**Passing collection objects by value forces C++ to copy the entire object, which is inefficient. Passing by reference also makes it possible for a function to change the value of the collection.**

15. What declaration would you use to initialize a variable called **chessboard** to an 8×8 grid, each of whose elements is a character?

Grid<char> chessboard = new Grid<char>(8, 8);

16. Given the **chessboard** variable from the preceding exercise, how would you assign the character **'R'** (which stands for a white rook in standard chess notation) to the squares in the lower left and lower right corners of the board?

chessboard[0][0] = chessboard[0][7] = 'R';

17. What do the acronyms *LIFO* and *FIFO* stand for? How do these terms apply to stacks and queues?

***LIFO* stands for “last in, first out”; *FIFO* stands for “first in, first out.”**

18. What are the names of the two fundamental operations for a stack?

**push and pop*.***

19. What are the names for the corresponding operations for a queue?

**enqueue and dequeue*.***

20. What does the **peek** operation do in each of the **Stack** and **Queue** classes?

**The peek operation looks at the top element of a stack or the head element of a queue without removing it from the collection.**

21. Describe in your own words what is meant by the term *discrete time* in the context of a simulation program.

**Simulations typically divide time into discrete units, each of which is short enough to ensure that multiple events are unlikely to occur.**

22. What are the two type parameters used with the **Map** class?

**The type of the key and the type of the value.**

23. What happens if you call **get** for a key that doesn’t exist in a map?

**The get method returns the default value for the key type.**

24. What are the syntactic shorthand forms for **get** and **put** that allow you to treat maps as associative arrays?

**The call map[*key*] is equivalent to map.get(*key*); the assignment statement map[*key*] = *value* is equivalent to map.put(*key*, *value*);**

25. Why do the Stanford libraries include a separate **Lexicon** class even though it is easy to implement a lexicon using the **Set** class?

**The Lexicon class is faster and more efficient in its use of space.**

26. What are the two kinds of data files supported by the constructor for the **Lexicon** class?

**1. Text files containing words arranged line by line**

**2. A binary data format that stores the lexicon in a compact form**

27. What is the general form of the range‑based **for** loop pattern?

for (*type var* : *collection*) . . .

28. What reason does the chapter offer for disallowing the use of the range‑based **for** loop with the **Stack** and **Queue** classes?

**The range‑based for violates the integrity of the Stack and Queue classes by giving clients access to elements other than the one at the top of the stack or the head of the queue.**

29. Describe the order in which the range‑based **for** loop processes elements for each of the collection classes introduced in this chapter.

**• The Vector class processes elements in index order.**

**• The Grid class processes elements in row-major order.**

**• The Map class processes elements in the natural order for the key type.**

**• The Lexicon and Set classes process elements in the natural order for the value type.**

**• The HashMap and HashSet classes process elements in a seemingly random order.**