**CS2\_ Exception Handling Review Questions:**

1. What is a throw point?

2. What is an exception handler?

3. Explain the difference between a try block and a catch block.

4. What happens if an exception is thrown, but not caught?

5. What is “unwinding the stack”?

6. What happens if an exception is thrown by a class’s member function?

7. How do you prevent a program from halting when the new operator fails to allocate memory?

8. Why is it more convenient to write a function template than a series of overloaded functions?

9. Why must you be careful when writing a function template that uses operators such as [] with its parameters?

10. What is a container? What is an iterator?

11. What two types of containers does the STL provide?

12. What STL algorithm randomly shuffles the elements in a container?

**Fill-in-the-Blank**

13. The line containing a throw statement is known as the \_\_\_\_\_\_\_\_\_\_.

14. The \_\_\_\_\_\_\_\_\_\_ block contains code that directly or indirectly might cause an exception to be thrown.

15. The \_\_\_\_\_\_\_\_\_\_ block handles an exception.

16. When writing function or class templates, you use a(n) \_\_\_\_\_\_\_\_\_\_ to specify a generic data type.

17. The beginning of a template is marked by a(n) \_\_\_\_\_\_\_\_\_\_.

18. When defining objects of class templates, the \_\_\_\_\_\_\_\_\_\_ you wish to pass into the type parameter must be specified.

19. A(n) \_\_\_\_\_\_\_\_\_\_ template works with a specific data type.

20. A(n) \_\_\_\_\_\_\_\_\_\_ container organizes data in a sequential fashion similar to an array.

21. A(n) \_\_\_\_\_\_\_\_\_\_ container uses keys to rapidly access elements.

22. \_\_\_\_\_\_\_\_\_ are pointer-like objects used to access data stored in a container.

23. The \_\_\_\_\_\_\_\_\_\_ exception is thrown when the new operator fails to allocate the requested amount of memory.

**Algorithm Workbench**

24. Write a function that searches a numeric array for a specified value. The function should return the subscript of the element containing the value if it is found in the array. If the value is not found, the function should throw an exception.

25. Write a function that dynamically allocates a block of memory and returns a char pointer to the block. The function should take an integer argument that is the amount of memory to be allocated. If the new operator cannot allocate the memory, the function should return a null pointer.

26. Make the function you wrote in Question 24 a template.

27. Write a template for a function that displays the contents of an array of any type.

28. A program has the following definition statements:

vector<int> numbers;

vector<int>::iterator iter;

Write code that uses the iterator to display all the values stored in the vector.

29. Write a statement that performs the STL binary\_search algorithm on the vector

defined in Question 28.

30. A program has the following definition:

vector<double> numbers;

The same program also has the following function:

void display(double n)

{

cout << n << endl;

}

Write code that uses the STL for\_each algorithm to display the elements of the numbers vector using the display function.

**True or False**

31. T F There can be only one catch block in a program.

32. T F When an exception is thrown, but not caught, the program ignores the error.

33. T F Data may be passed with an exception by storing it in members of an exception class.

34. T F Once an exception has been thrown, it is not possible for the program to jump back to the throw point.

35. T F All type parameters defined in a function template must appear at least once in the function parameter list.

36. T F The compiler creates an instance of a function template in memory as soon as it encounters the template.

37. T F A class object passed to a function template must overload any operators used on the class object by the template.

38. T F Only one generic type may be used with a template.

39. T F In the function template definition, it is not necessary to use each type parameter declared in the template prefix.

40. T F It is possible to overload two function templates.

41. T F It is possible to overload a function template and an ordinary (nontemplate) function.

42. T F A class template may not be derived from another class template.

43. T F A class template may not be used as a base class.

44. T F Specialized templates work with a specific data type.

45. T F When defining an iterator from the STL, the compiler automatically creates the right kind, depending upon the container it is to be used with.

46. T F STL algorithms are implemented as function templates.

**Find the Error**

Each of the following declarations or code segments has errors. Locate as many as possible.

47.

catch

{

quotient = divide(num1, num2);

cout << "The quotient is " << quotient << endl;

}

try (string exceptionString)

{

cout << exceptionString;

}

48. try

{

quotient = divide(num1, num2);

}

cout << "The quotient is " << quotient << endl;

catch (string exceptionString)

{

cout << exceptionString;

}

49.

template <class T>

T square(T number)

{

return T \* T;

}

50. template <class T>

int square(int number)

{

return number \* number;

}

51.

template <class T1, class T2>

T1 sum(T1 x, T1 y)

{

return x + y;

}

52. Assume the following definition appears in a program that uses the SimpleVector class template presented in this chapter.

int <SimpleVector> array(25);

53. Assume the following statement appears in a program that has defined valueSet as an object of the SimpleVector class presented in this chapter. Assume that valueSet is a vector of int s, and has 20 elements.

cout << valueSet<int>[2] << endl;