IS0020 Program Design and Software Tools
Spring Term, 2004
Final Examination, April 20

Name:

Instruction

There are two parts in this test. The first part is worth 80 points. The second part constitutes 20 points and there are *five* questions, but you have to attempt only *three*. There are also some bonus points you can get. Maximum time allowed is 2 hours and 30 minutes.

Good Luck!

Part I:

A. Fill in the blanks in each of the following statements [Score: 20]:

1. A base class's _________________________ members can be accessed only in the base-class definition or in derived-class definitions.
2. In a(n) _________________________ relationship, a class object has one or more objects of other classes as members.
3. In single inheritance, a class exists in a(n) _________________________ relationship with its derived classes.
4. _________________________ inheritance can be used as an alternative to composition.
5. A base class's _________________________ members are accessible anywhere that the program has a handle to an object of that base class or to an object of one of its derived classes.
6. C++ provides for _________________________, which allows a derived class to inherit from many base classes, even if these base classes are unrelated.
7. When an object of a derived class is instantiated, the base class's _________________________ is called implicitly or explicitly to do any necessary initialization of the base-class data members in the derived-class object.

8. When deriving a class from a base class with **protected** inheritance, public members of the base class become ________________________ members of the derived class, and protected members of the base class become ________________________ members of the derived class.

9. Classes from which objects can be instantiated are called __________________________________ classes.

10. _______________________ operator can be used to downcast base-class pointers safely.

11. Operator typeid returns a reference to a(n) _______________________ object.

12. ________________________ involves using a base-class pointer or reference to invoke virtual functions on base-class and derived-class objects.

13. Casting a base-class pointer to a derived-class pointer is called _______________________.

14. Templates enable us to specify, with a single code segment, an entire range of related functions called ________________________, or an entire range of related classes called ________________________.

15. The related functions generated from a function template all have the same name, so the compiler uses ________________________ resolution to invoke the proper function.

16. Class templates also are called ______________________ types.

17. The operator is used with a class-template name to tie each member function definition to the class template's scope.

18. As with **static** data members of non-template classes, **static** data members of class-template specializations must also be initialized at ______________________ scope.

19. The two types of STL containers are ______________________ containers and ______________________ containers.

20. The three STL adapters are ______________________, ______________________, and ______________________.

B. Multiple Choice Questions [Score: 60]:

1. Assume we have a base class `Shape` and derived classes `Triangle` and `Circle`. Which of the following member functions should be `virtual`?
   (a) `radius`
   (b) `calculateArea`
   (c) `perimeter`
   (d) both (b) and (c)

2. The main difference between a pure virtual function and a virtual function is
   (a) the return type
   (b) a pure virtual function cannot have an implementation.
   (c) the inheritance properties
   (d) the location in the class.

3. Which statement is *false* about dynamic binding?
   (a) It allows independent software vendors to hide proprietary secrets.
   (b) It eliminates the usefulness of separate header and source files.
   (c) It allows software developers to derive new classes compatible with existing software.
   (d) The program chooses the correct functions at execution time, rather than compile time.

4. Which of the following is *false*?
   (a) with a non-template class, one copy of a `static` data member is shared among all objects created from that class
   (b) each template class instantiated from a class template has its own copy of each `static` data member
   (c) `static` data members of both template and non-template classes need to be initialized at file scope
   (d) `static` member functions are shared between each class-template specialization in the class template.

5. For a non-empty linked list, select the code that should appear in a function that adds a node to the end of the list. `newPtr` is a pointer to the new node to be added, and `lastPtr` is a pointer to the current last node. Each node contains a pointer `nextPtr`, a link to a node.
   (a)
   ```
   lastPtr->nextPtr = newPtr;
   lastPtr = newPtr;
   ```
   (b)
   ```
   lastPtr = newPtr;
   lastPtr->nextPtr = newPtr;
   ```
   (c)
   ```
   newPtr->nextPtr = lastPtr;
   lastPtr = newPtr;
   ```
   (d)
   ```
   lastPtr = newPtr;
   newPtr->nextPtr = lastPtr;
   ```

6. Which of the following is most likely a base-class of the other three?
   (a) `automobile`
   (b) `convertible`
   (c) `miniVan`
   (d) `sedan`

7. Assuming the definition,
   ```
   class Circle : public Point
   ```

which of the following is false?
(a) the colon (**`:`**) in the header of the class definition indicates inheritance
(b) the keyword **`public`** indicates the type of inheritance
(c) All the **`public`** and **`protected`** members of class **`Circle`** are inherited as **`public`** and **`protected`** members, respectively, into class **`Point`**
(d) **`Point`** is the base class and **`Circle`** is the derived class

8. Suppose class **`A`** inherits from base class **`B.`** What is the order in which for the constructors and destructors will be called when an object of class **`A`** is instantiated then destroyed?
(a) **`B`** constructor, **`A`** constructor, **`A`** destructor, **`B`** destructor
(b) **`B`** constructor, **`A`** constructor, **`B`** destructor, **`A`** destructor.
(c) **`A`** constructor, **`B`** constructor, **`B`** destructor, **`A`** destructor.
(d) It depends on the order in which the object of class **`B`** appears on class **A**'s member initializer list.

9. **`Employee`** is a base class and **`HourlyWorker`** is a derived class, with an overridden **`print`** function. Given the following statements, will the output of the two **`print`** function calls be identical?
```
HourlyWorker h;
Employee *ePtr = &h;
ePtr->print();
ePtr->Employee::print();
```
(a) yes, if **`print`** is a **`virtual`** function.
(b) no, if **`print`** is a non-**`virtual`** function.
(c) yes, if **`print`** is non-**`virtual`** function.
(d) both (a) and (c)

10. The line
```
virtual double earnings() const = 0;
```
appears in the class definition. You cannot deduce that
(a) a derived class will override this method.
(b) the class is an abstract class.
(c) the class is probably generic.
(d) other classes are probably derived from this one.

11. Abstract classes
(a) contain only one pure **`virtual`** function.
(b) can have objects instantiated from them if the proper permissions are set.
(c) cannot have abstract derived classes.
(d) are defined, but the programmer never intends to instantiate any objects them.

12. Function-template specializations
(a) share one copy of the function template.
(b) are generated by the compiler.
(c) have a maximum allowed number of type parameters.
(d) are not more concise than the equivalent set of overloaded functions.

13. A function template can be overloaded by
(a) using other function templates with the same function name and parameters.
(b) using non-template functions with the same name and different parameters.

(c) using non-template functions with the same name and the same parameters.
(d) using other function templates with a different name but the same parameters.

14. Non-type parameters are
(a) unable to have default arguments
(b) specified before the type parameter
(c) **`const`**
(d) not optional for class templates

15. Which of the following is false?
(a) with a non-template class, one copy of a **`static`** data member is shared among all objects created from that class
(b) each template class instantiated from a class template has its own copy of each **`static`** data member
(c) **`static`** data members of both template and non-template classes need to be initialized at file scope
(d) **`static`** member functions are shared between each class-template specializaion in the class template.

16. Exception handling allows a program to
(a) terminate in a controlled manner
(b) be more robust and fault-tolerant
(c) continue executing as if no problem was encountered
(d) all of the above

17. Once an exception is thrown, when can control return to the **`throw`** point?
(a) never
(b) only if the exception was caught
(c) use braces **`{}`**
(d) always

18. The **`catch(...)`** statement is
(a) always executed regardless if an exception is thrown.
(b) only executed if it is the last **`catch`** statement and an exception is thrown and possibly caught by a previous **`catch`** handler in the same sequence.
(c) only executed if an exception is thrown and precedes the **`try`** block.
(d) always executed if it is the first in a list of **`catch`** blocks and an exception is thrown.

19. The correct order in which an exception is detected and handled is
(a) **`try`**, **`catch`**, **`throw`**
(b) **`throw`**, **`catch`**, **`try`**
(c) **`catch`**, **`throw`**, **`try`**
(d) **`try`**, **`throw`**, **`catch`**

20. The **`try`** block does not
(a) enclose the code that may throw the exception.
(b) enclose the **`catch`** block.
(c) test nested **`try`** blocks for additional **`catch`** statements
(d) have exceptions explicitly or implicitly **`thrown`** in the **`try`** block itself.

21. The purpose of stack unwinding is to
(a) attempt to **`catch`** exceptions that are not caught in their scope.
(b) improve **`catch`** blocks by allowing them to handle multiple exceptions.
(c) return control to the function that created the exception.
(d) aid the **`terminate`** command in shutting down the program.

22. If dynamic memory has been allocated for an object and an exception occurs, then
(a) the **`catch`** block will not work properly.
(b) a memory leak could result.
(c) the object's constructor will cause another exception.
(d) multiple pointers to memory could be created.

23. Which file open mode would be used to write data only to the end of an existing file?
(a) **`ios::app`**
(b) **`ios::in`**
(c) **`ios::out`**
(d) **`ios::trunc`**

24. Select the false statement
(a) C++ imposes no structure on a file.
(b) C++ files include information about their structure.
(c) The programmer must impose a structure on a file.
(d) C++ files do not understand notions such as "records" and "fields."

25. Which of the following is not a *disadvantage* of trying to modify a sequential access file?
(a) modifying data can potentially destroy other data.
(b) overwriting a record with another record of the same size is very difficult
(c) it may be necessary to modify every record in the file to make a slight change
(d) something which is stored in the same number of “raw data” bytes internally may not take up the same amount of space in a file.

26. What is not true about this code segment?
```
location = fileObject.tellg();
```
(a) **`tellg`** is a member function of **`fileObject`**.
(b) **`location`** is a pointer.
(c) the value of **`location`** must be less than or equal to the number of bytes in the file.
(d) **`fileObject`** is an **`istream`** object.

27. Random access files are more effective than sequential files for
(a) instant access to data
(b) updating data easily
(c) inserting data into the file without destroying other data
(d) all of the above

28. For an **`ifstream`** object **`inCredit`**, a **`class`** type **`clientData`** and a **`clientData`** structure **`client`**, the proper way read in one **`class`** is
(a) **`inCredit.read(&client, sizeof( clientData ) );`**
(b) **`inCredit.read( reinterpret_cast<char *>( &client ), sizeof( clientData ) );`**
(c) **`inCredit.read( reinterpret_cast<char *>( &client ), clientData);`**

(d) `inCredit.read(char * ( &client ), sizeof( clientData ) );`

29. Select the proper object type.
_________ `file("file.dat", ios::in | ios::out);`
(a) `iostream`
(b) `fstream`
(c) `ofstream`
(d) `ifstream`

30. Select the false statement. When objects are saved to a file
(a) only data is returned about the object, not type information.
(b) they can be read from the file at a later time.
(c) different types of objects cannot be written to the same file.
(d) it is often necessary to output the object's type as well.

31. Given the line
```
      delete newPtr;
```
what can you conclude?
(a) The memory referenced by `newPtr` is released only if it is needed by the system.
(b) `newPtr` is of type `void *`.
(c) `newPtr` only exists if there was an error freeing the memory.
(d) `newPtr` still exists.

32. __________ is not an advantage of linked lists when compared to arrays.
(a) Dynamic memory allocation
(b) Efficient insertion and deletion
(c) Direct access to any list element
(d) Efficient use of memory

33. Iterators are similar to pointers because of the
(a) `*` and `++` operators.
(b) `->` operator.
(c) `begin` and `end` functions.
(d) `&` operator.

34. Which of the following is a difference between vectors and arrays?
(a) access to any element using the [] operator
(b) contiguous blocks of memory
(c) the ability to change size dynamically
(d) efficient, direct-access

35. Which of the following is not a sequence container provided by the STL?
(a) `vector`
(b) `array`
(c) `list`
(d) `deque`

36. Class `deque` provides
(a) efficient indexed access to its elements
(b) the ability to add storage at either end of the deque

(c) efficient insertion and deletion operations at its front and back
(d) all of the above

37. The main difference between **`set`** and **`multiset`** is
(a) their interface.
(b) that one deals with keys only, and the other deals with key/value pairs.
(c) their efficiency.
(d) how they handle duplicate keys.

38. The difference between maps and sets is that
(a) the elements of **`multimaps`** and **`maps`** are pairs of keys and values instead of individual values
(b) the elements of **`multisets`** and **`sets`** are pairs of keys and values instead of individual values
(c) **`multimaps`** and **`maps`** can be ordered using comparator functions and **`multisets`** and **`sets`** can not
(d) **`multisets`** and **`sets`** can be ordered using comparator functions and **`multimaps`** and **`maps`** can not

39. Select the false statement. *Container adapters*
(a) do not provide the actual data structure implementation for elements to be stored.
(b) have an underlying data structure.
(c) can all **`push`** and **`pop`** elements.
(d) have limited iterator support.

40. A queue receives the following commands (in pseudo-code):
enqueue 4, 6, 8, 3, 1 *(4 is the first element to be enqued)*
dequeue 3 elements
enqueue 3, 1, 5, 6
dequeue 2 elements
What number is at the front of the queue?
(a) 3
(b) 4
(c) 5
(d) 6

Part II

Each question is worth 7 points

Note that you need to attempt only *three* out of *five* questions. However, you can get 7 bonus points for one more question.

1. Write a function template called sortElements for sorting an array of objects. Use either *bubble* sort or *selection* sort to sort the elements. Your function should be able to be used with an array of int or double elements. Suppose in future you want to use this function template on an array of objects of a newly defined class called BigInteger, which allows numbers of arbitrary sizes to be stored. Note that both int and double have limitation on the maximum value that can be stored because of the fixed byte size associated with them. BigInteger essentially removes the byte-size limitation of int. To be able to reuse sortElements on an array of BigIntegers, enumerate things that need to be kept in mind when defining the class BigInteger.

2. Suppose a class template has the header

```
template< class T1 > class C1
```

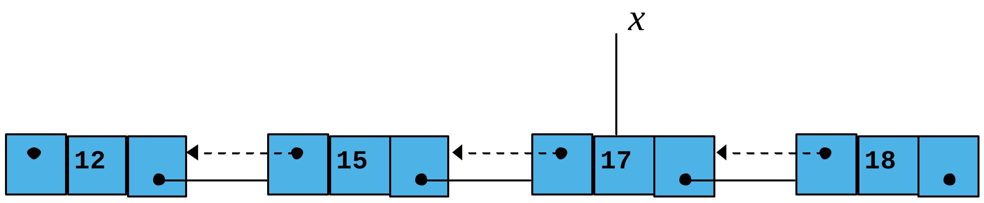
Describe the friendship relationships established by placing each of the following friendship declarations inside this class-template header. Identifiers beginning with "f" are functions, identifiers beginning with "C" are classes and identifiers beginning with "T" can represent any type (i.e., built-in types or class types).

a. `friend void f2( C1< T1 > &);`
b. `friend void C2::f4();`
c. `friend void C3< T1 >::f5( C1< T1 > & );`

3. Consider the Point class that we studied in class, shown below. Suppose we want to define two other classes:Square and Cube. Would you use *inheritance* or *composition* to define the new classes – give reasons. Define your Square and Cube classes. Include area and volume related class specific information and behavior in the new classes.

```
class Point {

public:
   Point3( int = 0, int = 0 ); // default constructor

   void setX( int );    // set x in coordinate pair
   int getX() const;    // return x from coordinate pair
   void setY( int );    // set y in coordinate pair
   int getY() const;    // return y from coordinate pair
   void print() const; // output Point3 object

private:
   int x;  // x part of coordinate pair
   int y;  // y part of coordinate pair

}; // end class Point3
```

4. Consider a linked list as shown below. Assume leftPtr (dotted arrow line) and rightPtr are pointer fields in each node. Write a code block to
   a. Insert a node to the left of the node pointed to by variable *x* (between 15 and 17), and
   b. delete the node pointed to by variable *x*.



5. Give reasons for or against the following statement: *The Standard Template Library defines powerful, template-based, reusable components for generic programming using three key components – containers, iterators and algorithms*.