CHAPTER 10

Fill in the blanks in each of the following:

a) **Initializer list** must be used to initialize constant members of a class.

b) A nonmember function must be declared as a(n) **Friend** of a class to have access to that class's private data members.

c) A constant object must be **Initialized** ; it cannot be modified after it's created.

d) A(n) **Static** data member represents class-wide information.

e) An object's non-static member functions have access to a "self pointer" to the object called the **This** pointer.

f) Keyword **constant** specifies that an object or variable is not modifiable.

g) If a member initializer is not provided for a member object of a class, the object's **default constructor** is called

i) Member objects are constructed **Before** their enclosing class object

chapter 11

Fill in the blanks in each of the following:

a) Suppose a and b are integer variables and we form the sum a + b. Now suppose c and d are floating-point variables and we form the sum c + d. The two + operators here are clearly being used for different purposes. This is an example of **operator overloading**.

b) Keyword **operator** introduces an overloaded-operator function definition.

c) To use operators on class objects, they must be overloaded, with the exception of oper-ators **=(assignment)** and **&(address)**, **,(comma)**

d) The **precedence, associativity, and "arity** of an operator cannot be changed by overload-ing the operator.

e) The operators that cannot be overloaded are **(.), (?:), (.*), and (::)**

.

f) The **delete** operator reclaims memory previously allocated by new.

g) The **new** operator dynamically allocates memory for an object of a specified type and returns a(n) **pointer** to that type.

Chapter 12

Fill in the blanks in each of the following statements:

a) **inheritance** is a form of software reuse in which new classes absorb the data and behaviors of existing classes and embellish these classes with new capabilities.

b) A base class's **protected** members can be accessed in the base-class definition, in derived-class definitions and in friends of the base class its derived classes.

c) In a(n) **is-a or inheritance** relationship, an object of a derived class also can be treated as an object
of its base class.

d) In a(n) **has-a or composition or aggregation** relationship, a class object has one or more objects of other classes as members.

e) In single inheritance, a class exists in a(n) **hierarchical** relationship with its derived classes.

f) A base class's **public** members are accessible within that base class and anywhere that the program has a handle to an object of that class or one of its derived classes.

g) A base class's protected access members have a level of protection between those of public and **private** access.

h) C++ provides for **multiple inheritance** which allows a derived class to inherit from many base classes, even if the base classes are unrelated.

i) When an object of a derived class is instantiated, the base class's **constructor** is called implicitly or explicitly to do any necessary initialization of the base-class data members in

the derived-class object.

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

k) When deriving a class from a base class with protected inheritance, public members of the base class become **protected** members of the derived class, and protected members of the base class become **protected** members of the derived class

State whether each of the following is true or false. If false, explain why.

a) Base-class constructors are not inherited by derived classes.**(True)**

b) A has-a relationship is implemented via inheritance.**(False)**

c) A Car class has an is-a relationship with the SteeringWheel and Brakes classes. **(False)**

d) Inheritance encourages the reuse of proven high-quality software. **(True)**

e) When a derived-class object is destroyed, the destructors are called in the reverse order of the constructors.. **(True)**

chapter 13

Fill in the blanks in each of the following statements:

a) Treating a base-class object as a(n) **derived class object** can cause errors.

b) Polymorphism helps eliminate **switch** logic.

c) If a class contains at least one pure virtual function, it's a(n) **abstract** class.

d) Classes from which objects can be instantiated are called **concrete** classes.

e) Operator **dynamic cast** can be used to downcast base-class pointers safely.

f) Operator typeid returns a reference to a(n) **type info** object.

g) **polymorphism** involves using a base-class pointer or reference to invoke virtual functions on base-class and derived-class objects.

h) Overridable functions are declared using keyword **virtual** .

i) Casting a base-class pointer to a derived-class pointer is called **downcasting**

State whether each of the following is true or false. If false, explain why.

a) All virtual functions in an abstract base class must be declared as pure virtual functions.**(false)**

b) Referring to a derived-class object with a base-class handle is dangerous. **(false)**

c) A class is made abstract by declaring that class virtual. **(false)**

d) If a base class declares a pure virtual function, a derived class must implement that function to become a concrete class.**(true)**

e) Polymorphic programming can eliminate the need for switch logic**(true)**