CENTENNIAL COLLEGE

LAB 4

COMP-217 SEC- 001

Name: Muhammad Saad Student Id: 301077320

Instructions:

* This assignment is supposed to be completed individually.
* Add you name and student Id on top of the Lab 4 Doc.
* Please add correct answer under answer section for each question.
* Submit the file with correct responses in the submission box (LAB 4) under Assignments section.
* Each question has 1 mark each.
* Please pick up the answers as a/b/c/d from options (do not add complete text).
* Please add the answers sequentially as per the questions by editing this file.
* For submission name the file with following convention: **studentfirstname\_lastname\_lab4**

Q1: Member access specifiers (public and private) can appear:

a. In any order and multiple times.

b. In any order (public first or private first) but not multiple times.

c. In any order and multiple times, if they have brackets separating each type.

d. Outside a class definition.

**Ans: B**

Q2: Which of the following preprocessor directives does *not* constitute part of the preprocessor wrapper?

a. #define

b. #endif

c. #ifndef

d. #include

**Ans: B**

Q3: Member function definitions:

a. Always require the scope resolution operator (::).

b. Require the scope resolution operator only when being defined outside of the definition of their class.

c. Can use the scope resolution operator anywhere, but become public functions.

d. Must use the scope resolution operator in their function prototype.

**Ans: B**

Q4: Parameterized stream manipulator setfill specifies the fill character that’s displayed when an output is displayed in a field wider than the number of characters or digits in the output. The effect of setfill applies:

a. Only to the current value being displayed.

b. Only to outputs displayed in the current statement.

c. Until explicitly set to a different setting.

d. Until the output buffer is flushed.

**Ans: C**

Q5: Every object of the same class:

a. Gets a copy of every member function and member variable.

b. Gets a copy of every member variable.

c. Gets a copy of every member function.

d. Shares pointers to all member variables and member functions.

**Ans: D**

Q6: A class’s functions can throw exceptions, such as \_\_\_\_\_\_\_\_\_\_to indicate invalid data.

a. invalid\_data

b. bad\_data

c. invalid\_argument

d. bad\_argument

**Ans: C**

Q7: Which of the following statements is *false* (assume we're referring to class Time)?

a. Often a class’s interface and implementation will be created and compiled by one programmer and used by a separate programmer who implements the client code that uses the class.

b. To hide the class’s member-function implementation details, the class-implementation programmer would provide the client-code programmer with the header Time.h (which specifies the class’s interface and data members) and the Time object code (i.e., the machine-code instructions that represent Time’s member functions).

c. The client-code programmer is not given Time.cpp, so the client remains unaware of how Time’s member functions are implemented.

d. The client-code programmer needs to know only Time’s interface to use the class and must be able to link its object code.

**Ans: B**

Q8: Which of the following statements is false (assume we're referring to class Time)?

a. Since the interface of the class is part of the class definition in the Time.h header, the client-code programmer must have access to this file and must #include it in the client’s source-code file.

b. When the client code is compiled, the compiler uses the class definition in Time.h to ensure that the main function creates and manipulates objects of class Time correctly.

c. The linker’s output is the executable Time application that users can execute to create and manipulate a Time object.

d. Compilers and IDEs typically invoke the linker for you after compiling your code.

**Ans: B**

Q9: Variables defined inside a member function of a class have:

a. File scope.

b. Class scope.

c. Block scope.

d. Class or block scope, depending on whether the binary scope resolution operator (::) is used.

**Ans: C**

Q10: A class-scope variable hidden by a block-scope variable can be accessed by preceding the variable name with the class name followed by:

a. ::

b. :

c. .

d. ->

**Ans: A**

Q11: The type of function a client would use to check the balance of a bank account would be:

a. A utility function.

b. A predicate function.

c. An access function.

d. A constructor.

**Ans: C**

Q12: Utility functions:

a. Are private member functions that support operations of the class’s other member functions.

b. Are part of a class’s interface.

c. Are intended to be used by clients of a class.

d. Are a type of constructor.

**Ans: A**

Q13: A default constructor:

a. Is a constructor that must receive no arguments.

b. Is the constructor generated by the compiler when no constructor is provided by the programmer.

c. Does not perform any initialization.

d. Both (a) and (b).

**Ans: B**

Q14: If a member function of a class already provides all or part of the functionality required by a constructor or another member function then:

a. Copy and paste that member function’s code into this constructor or member function.

b. Call that member function from this constructor or member function.

c. That member function is unnecessary.

d. This constructor or member function is unnecessary.

**Ans: B**

Q15Assuming the following constructor is provided for class Time

explicit Time(int = 0, int = 0, int = 0);

which of the following is *not* a valid way to initialize a Time object?

a. Time t1;

b. Time t2{22, 40};

c. Time t3(22, 40);

d. a), b) and c) are all valid ways to initialize a Time object.

**Ans: D**

Q16Which of the following statements is *false*?

a. You can overload a classes constructors.

b. There is no mechanism in C++ for a constructor to call another constructor in the same class.

c. Just as a constructor can call a class’s other member functions to perform tasks, C++11 allows constructors to call other constructors in the same class.

d. To overload a constructor, provide in the class definition a prototype for each version of the constructor, and provide a separate constructor definition for each overloaded version.

**Ans: B**

Q17: Which of the following is *not* true of a constructor and destructor of the same class?

a. They both have the same name aside from the tilde (~) character.

b. They are both usually called once per object created.

c. They both are able to have default arguments.

d. Both are called automatically, even if they are not explicitly defined in the class.

**Ans: C**

Q18: Which of the following is *not* true of a destructor?

a. It performs termination housekeeping.

b. It is called before the system reclaims the object’s memory.

c. If the programmer does not explicitly provide a destructor, the compiler creates an “empty” destructor.

d. It releases the object’s memory.

**Ans: B**

Q19: Given the class definition:

class CreateDestroy

{

public:

CreateDestroy() {cout << "constructor called, ";}

~CreateDestroy() {cout << "destructor called, ";}

};

What will the following program output?

int main()

{

CreateDestroy c1;

CreateDestroy c2;

return 0;

}

a. constructor called, destructor called, constructor called, destructor called,

b. constructor called, destructor called,

c. constructor called, constructor called,

d. constructor called, constructor called, destructor called, destructor called,

**Ans: D**

Q20: Given the class definition:

class CreateDestroy

{

public:

CreateDestroy() {cout << "constructor called, ";}

~CreateDestroy() {cout << "destructor called, ";}

};

What will the following program output?

int main()

{

for (int i = 1; i <= 2; ++i) {

CreateDestroy cd;  
 }

return 0;

}

a. constructor called, destructor called, constructor called, destructor called,

b. constructor called, constructor called,

c. constructor called, constructor called, destructor called, destructor called,

d. Nothing.

**Ans: C**

Q21: Returning references to non-const, private data:

a. Allows private functions to be modified.

b. Is only dangerous if the binary scope resolution operator (::) is used in the function prototype.

c. Allows private member variables to be modified, thus “breaking encapsulation.”

d. Results in a compiler error.

**Ans: C**

Q22: A client changing the values of private data members is:

a. Only possible by calling private member functions.

b. Possible using public functions and references.

c. Never possible.

d. Only possible if the private variables are not declared inside the class.

**Ans: B**

Q23: The assignment operator (=) *can* be used to:

a. Test for equality.

b. Copy data from one object to another.

c. Compare two objects.

d. Copy a class.

**Ans: B**

Q24: Which of the following statements will *not* produce a syntax error?

e. Defining a const member function that modifies a data member of the object.

f. Invoking a non-const member function on a const object.

g. Declaring an object to be const.

h. Declaring a constructor to be const.

**Ans: G**

Q25: The code fragment:

Increment::Increment(int c, int i)

: increment (i)

{

count = c;

}

does *not* cause any compilation errors. This tells you that:

a. count must be a non-const variable.

b. count must be a const variable.

c. increment must be a non-const variable.

d. increment must be a const variable.

**Ans: A**

Q26: When composition (one object having another object as a member) is used:

a. The host object is constructed first and then the member objects are placed into it.

b. Member objects are constructed first, in the order they appear in the host constructor’s initializer list.

c. Member objects are constructed first, in the order they are declared in the host’s class.

d. Member objects are destructed last, in the order they are declared in the host’s class.

**Ans: B**

Q27: An error occurs if:

a. A non-reference, non-const, primitive data member is initialized in the member initialization list.

b. An object data member is not initialized in the member initialization list.

c. An object data member does not have a default constructor.

d. An object data member is not initialized in the member initialization list and does not have a default constructor.

**Ans: D**

Q28: If the line:

friend class A;

appears in class B, and the line:

friend class B;

appears in class C, then:

a. Class A is a friend of class C.

b. Class A can access private variables of class B.

c. Class C can call class A’s private member functions.

d. Class B can access class A’s private variables.

**Ans: A**

Q29: Which of the following statements about friend functions and friend classes is false?

a. A class can either grant friendship to or take friendship from another class using the friend keyword.

b. A friend declaration can appear *anywhere* in a class definition.

c. A friend of a class can access all of its private data member and member functions.

d. The friendship relationship is neither symmetric nor transitive.

**Ans: D**

Q30: For a *non-constant* member function of class Test, the this pointer has type:

a. const Test \*

b. Test \* const

c. Test const \*

d. const Test \* const

**Ans: B**

Q31: Inside a function definition for a member function of an object with data member x, which of the following is *not* equivalent to this->x:

a. \*this.x

b. (\*this).x

c. x

d. None of the above are equivalent.

**Ans: A**

Q32: Assume that t is an object of class Test, which has member functions a(), b(), c() and d(). If the functions a(), b() and c() all return references to an object of class Test (using the dereferenced this pointer) and function d() returns void, which of the following statements will *not* produce a syntax error:

a. t.a().b().d();

b. a().b().t;

c. t.d().c();

d. t.a().t.d();

**Ans: C**

Q33: If Americans are objects of the same class, which of the following attributes would most likely be represented by a staticvariable of that class?

a. Age.

b. The President.

c. Place of birth.

d. Favorite food.

**Ans: B**

Q34: static data members of a certain class:

a. Can be accessed only if an object of that class exists.

b. Cannot be changed, even by objects of the same *that* class.

c. Have class scope.

d. Can only be changed by static member functions.

**Ans: C**

Q35: static member functions:

a. Can use the this pointer.

b. Can access only other static member functions and static data members.

c. Cannot be called until an object of their class is instantiated.

d. Can be declared const as well.

**Ans: B**

Q36: Polymorphism is implemented via:

a. Member functions.

b. virtual functions and dynamic binding.

c. inline functions.

d. Non-virtual functions.

**Ans: B**

Q37: Which of the following statements about polymorphism is *false*?

a. With polymorphism, you can direct a variety of objects to behave in manners appropriate to those objects without even knowing their types.

b. With polymorphism, new types of objects that can respond to existing messages can easily be incorporated into a system without modifying the base system.

c. Polymorphism enables you to deal in specifics and let the execution-time environment concern itself with the generalities.

d. To get polymorphic behavior among existing objects, those objects must be instantiated from classes in the same inheritance hierarchy.

**Ans: D**

Q38: Which of the following would not be a member function that derived classes Fish, Frog and Bird should inherit from base class Animal and then provide their own definitions for, so that the function call can be performed polymorphically?

a. eat

b. sleep

c. move

d. flapWings

**Ans: D**

Q39: Employee is a base class and HourlyWorker is a derived class, with a redefined non-virtual 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.

b. Yes, if print is a static function.

c. No.

d. It would depend on the implementation of the print function.

**Ans: C**

Q40: Which of the following assignments would be a compilation error?

a. Assigning the address of a base-class object to a base-class pointer.

b. Assigning the address of a base-class object to a derived-class pointer.

c. Assigning the address of a derived-class object to a base-class pointer.

d. Assigning the address of a derived-class object to a derived-class pointer.

**Ans: B**

Q41: Downcasting enables:

a. A derived-class object to be treated as a base-class object.

b. A base-class object to be treated as a derived-class object.

c. Making a base-class pointer into a derived-class pointer.

d. Making a derived-class pointer into a base -class pointer.

**Ans: D**

Q42: If objects of all the classes derived from the same base class all need to draw themselves, the draw function would most likely be declared:

a. private

b. virtual

c. protected

d. friend

**Ans: B**

Q43: virtual functions must:

a. Be overridden in every derived class.

b. Be declared virtualin every derived class.

c. Be declared virtual in the base class.

d. Have the same implementation in *every* derived class.

**Ans: C**

Q44: Which of the following statements about virtual functions is *false*?

a. They allow the program to select the correct implementation at execution time.

b. They can use either static or dynamic binding, depending on the handles on which the functions are called.

c. They do not remain virtual down the inheritance hierarchy.

d. They can be called using the dot operator.

**Ans: D**

Q45[C++11]: Which of the following statements is *true*?

a. In C++11, all classes can be used as base classes.

b. In C++11, only classes that are not declared as final can be used as base classes.

c. In C++11, only classes that are declared as base can be used as base classes.

d. None of the above

**Ans: B**

Q46: virtual destructors *must* be used when:

a. The constructor in the base class is virtual.

b. delete is used on a base-class pointer to a derived-class object.

c. delete is used on a derived-class object.

d. A constructor in either the base class or derived class is virtual.

**Ans: B**

Q47[C++11]: To help prevent errors, apply C++11’s \_\_\_\_\_\_\_\_ keyword to the prototype of every derived-class function that overrides a base-class virtual function.

a. virtual

b. abstract

c. overridable

d. None of the above.

**Ans: B**

Q48[C++11] In C++11, you can tell the compiler to explicitly generate the default version of a default constructor, copy constructor, move constructor, copy assignment operator, move assignment operator or destructor by following the special member function’s prototype with \_\_\_\_\_\_\_\_.

a. default

b. explicit

c. (default)

d. = default

**Ans: D**

Q49: Problems using switch logic to deal with many objects of different types do *not* include:

a. Forgetting to include an object in one of the cases.

b. Having to update the switch statement whenever a new type of object is added.

c. Having to track down every switch statement to do an update of object types.

d. Not being able to implement separate functions on different objects.

**Ans: D**

Q50: The line:

virtual double earnings() const = 0;

appears in a class definition. You *cannot* deduce that:

a. All classes that directly inherit from this class will *override* this method.

b. This class is an abstract class.

c. Any concrete class derived from this class will have an earnings function.

d. This class will probably be used as a base class for other classes.

**Ans: C**