OOP Practice Exam – Solutions

MCQs:

1.B 2.A 3.B 4.D 5.C 6.A 7.A 8.B 9.D 10.C

Short Questions:

Q1(a) It is a reference to a pointer.

Q1(b) ref->display()

Q2. Similarity: Both private and protected members cannot be accessed by functions outside of the class.

Differences:

1. Private members can be accessed by friend functions but protected cannot.

2. Protected members can be accessed by derived classes but private members cannot.

Q3(a) **const** objects cannot be updated, i.e. no data member can be changed.

(b) No, const objects can only be created for classes where all member functions are declared const.

Q4(a) Test(int a, int b, &ref, int p): b(b), cRef(ref), Part(p) {this->a = a; }

Q4(b)

char c = ‘a’;

char& ref = c;

Test t(5, 6, ref, 7);

Q5(a) They can be accessed without any object of the class. So they are used to maintain a single copy of some data that is common to all class members. This is efficient because only one variable is being maintained in memory instead of multiple copies (one with each object).

Q5(b) In the following class, the integer count is static:

class Computer{

private:

static int count;

public:

static int getCount() {return count;}

};

int Computer::count = 0;

int main()

{

cout<<Computer::getCount();

return 0;

}

Q5(c) Static functions are needed when you need to invoke any functionality of the class without actually creating an object. For example, one scenario requiring static functions is when you need to manipulate static class members, such as to get the value of count variable in the above code (in the answer to part (b)), you need a static function that can be called without any object.

Q6(a) It will give an error.

Q6(b) The expression int total = e1 + e2 + e3 is evaluated left to right, step by step. In the first step, e1 and e2 are added together, and the result is added into e3. When we add e1 and e2 together, the overloaded operator+ function is called, which returns an int. At the next step, this int needs to be added into e3. However, there is no function that can add an int and an employee together. Thus, you get an error at this point. The solution is to return an Employee, not an int, from the operator+ function.

Q7(a) This calls:

Base, Derived1, Base, Derived2, Derived3.

Q7(b)

Base, Derived 1, Derived 2, Derived3 (because now Derived1 and Derived2 inherit a single copy of Base, so the Base constructor is only called once).

Q8. Static polymorphism: function overloading

Dynamic polymorphism: When a base pointer points at a derived class object, and calls an overridden function of the derived class, the base pointer is bound to the derived class function at run-time.

Q9:

Rectangle R1: No, because Rectangle does not provide a definition for the pure virtual function it has inherited from Shape, and hence is itself an abstract class.

Shape s: No, it has a pure virtual function and is hence an abstract class.

Shape \*s: Yes, you can declare pointers of abstract classes.

Rectangle \*r: Yes, same reason as above.

BlueRectangle br: Yes, as it provides a concrete implementation of the pure virtual function that it has inherited from the Shape class.

Rectangle \*r = &br

Yes, a pointer of an abstract base class can point to an object of a concrete derived class.

Shape \*s = &br

Yes, same reason as above.

Q10:

1. Aggregation
2. Composition (deleting a shelf means all books that are part of it get deleted)
3. Association, as there is not a part whole relationship.
4. Aggregation (this is similar to how an address is related to a person)
5. Association (as there is a many to many relationship on both sides now).