CS 274—Object Oriented Programming with C++
Final Exam
December 16th

(2) 1. TRUE or FALSE: Namespaces allow you to have multiple memory locations with the same variable name.

(6) 2. Explain the error in the following groups of function declarations or write "No Error"

(a)
```
void f(int x, string y);
int f(int x, string y);
```

(b)
```
float g(int x, float y = 2.31, string z);
```

(c)
```
void h(int x);
void h();
```

(2) 3. Explain the difference between the keywords *struct* and *class*.

(6) 4. Consider the following class definition:

```
class TwoPoint{
        private:
                int x, y;
        public:
                void set(int x, int y);
};
```

Provide the definition of the method *set*. Write the method **outside** the class and **do not change** the prototype of the method.

(4) 5. Give two reasons to pass an object by reference.

(3) 6. Refer back to the TwoPoint class in #4 above. Write a default constructor for the class that **does not** have an empty parameter list.

(2) 7. Give one reason why a class would need a destructor.

(2) 8. Why can't the constant pointer *this* be used inside a *static* method?

(4) 9. Assume the necessary operators are overloaded:
(a) Write the following in operator form: b.operator=(a.operator+(c));
(b) Write the following in functional form: z = x + w*y;

(4) 10. Consider a class Complex in which addition has been overloaded. Let the variable *z* represent an object of type Complex. What two things need to be true in order for the expression *2 + z* to be legal?

(3) 11. Why can't << be overloaded as a method and still retain its standard functionality (ie, *cout* << *x*, where *x* is an object type)?

(2) 12. TRUE or FALSE: The assignment operator must be overloaded as a top-level function.

(4) 13. Consider the following inheritance hierarchy:

```
class A{
   protected:
         int x, y;
   public:
         int z;
}

class B:  public A{
   private:
        int a, b, c;
}
```

(a) How many data members does B have?
(b) How many of B's data members are *visible* in B?

(4) 14. Why is *protected* accessibility needed?

(3) 15. Give an example of a case where *virtual inheritance* is necessary (ie, a case where you need *virtual base classes* in a multiple inheritance hierarchy.)

(5) 16. Consider the following inheritance hierarchy:

```
class A{
   public:
        int x;
}

class B:  public A{
   private:
        int x;
   public:
         void set(int a, int b);
}
```

Write the definition for the method *set* that assigns the inputs *a* and *b* to B's data members.

(2) 17. TRUE or FALSE: If a base class has a default constructor, then none of the constructors in the derived class need to explicitly call a base class constructor.

(2) 18. TRUE or FALSE: If a base class has a parameterized constructor but no default constructor, then derived class objects cannot be instanced unless the derived class constructor explicitly calls one of the base class constructors.

(2) 19. TRUE or FALSE: If a base class has no constructors whatsoever, then a derived class object cannot be instanced.

(3) 20. With which type of binding is the vtable used?

(2) 21. TRUE or FALSE: Virtual inheritance is necessary for polymorphism.

(2) 22. What must you do if you want a class derived from an abstract base class to be non-abstract?

(2) 23. Template functions and template classes contain at least one class parameter (typically denoted by *T* in the text). What does this class parameter represent?

(2) 24. Fill in the blank: Container is to Iterator as Array is to __________________ .

(25) 25. Write a program that reads in test scores and applies two different curves to them. The program should

Contain a base class ScoreBank with two private data members: an integer pointer for the scores and a float for the average. The class should contain a method *EnterScores* which asks the user how many test scores are needed, allocates enough memory, and reads in the scores. The class should also contain a method *CalcAverage* which stores the average of the entered scores in the private float data member. Also have a *Output* function that prints a sorted list of test scores to the screen (you may use the *sort* algorithm from STL: see page 565) as well as the average. Be sure the class has an appropriate destructor.

Derive from ScoreBank a class Curve1 which contains a method *Curve*. This curve sets the average score to 75. Find out how far away from 75 the actual average is and then add this value to each test score. Overload the *Output* method to print, sorted, the original scores and the curved scores as well as the original and new average.

Derive from ScoreBank a class Curve2 which contains a method *Curve*. This curve sets the high score to 100 and scales the rest of the scores accordingly. Find the high score (you can use *sort* to do this) and divide all the scores by this value. Then overload the *Output* function to print the original scores, the new scores, and the averages for both sets.

Feel free to add any other methods or data members to the classes other than the ones I've mentioned if it will help you implement any of the required features.

Write a *main* function driver for these classes that creates objects of type Curve1 and Curve2 and asks the user to input a list of test scores into each object and then runs the *Output* functions from each object.