Name: _____

Username: _____

	Score	Max
I.		22
II.		24
III. p. 6		8
p. 7		12
IV. p. 8		6
p. 9		12
p. 10		12
p. 11		12
p. 12		12
Total		120

Instructions:

1. This is a closed-book, closed-notes exam.
2. You have 120 minutes for the exam.
3. Calculators, cell phones and laptops must be put away.
4. Clearly indicate your final answer.

I. Compiles/Not (1 points each)

For each question in this section, circle whether the line of code given would result in a compile-time error.

Questions 1–10 refer to the following declarations:

```
class SimpleClass {

public:
   SimpleClass() ;
   int pub1, pub2 ;
   void pubFunc(SimpleClass obj) ;

private:
   int priv1, priv2 ;
   void privFunc(const SimpleClass& constObj) ;
} ;
```

1. A statement in `pubFunc()` :

   ```
   pub1 = pub1 + pub2 ;
   ```

   COMPILES DOES NOT COMPILE

2. A statement in `pubFunc()` :

   ```
   priv1 = priv1 + priv2 ;
   ```

   COMPILES DOES NOT COMPILE

3. A statement in `privFunc()` :

   ```
   pub1 = pub1 + pub2 ;
   ```

   COMPILES DOES NOT COMPILE

4. A statement in `privFunc()` :

   ```
   priv1 = priv1 + priv2 ;
   ```

   COMPILES DOES NOT COMPILE

5. A statement in `pubFunc()` :

```
obj.pub1 = pub1 + pub2 ;
```

COMPILES DOES NOT COMPILE

6. A statement in `pubFunc()` :

```
obj.priv1 = priv1 + priv2 ;
```

COMPILES DOES NOT COMPILE

7. A statement in `privFunc()` :

```
constObj.pub1 = pub1 + pub2 ;
```

COMPILES DOES NOT COMPILE

8. A statement in `privFunc()` :

```
constObj.priv1 = priv1 + priv2 ;
```

COMPILES DOES NOT COMPILE

9. Statements in `main()`, :

```
SimpleClass A, B ;
A.pub1 = A.pub1 + B.pub2 ;
```

COMPILES DOES NOT COMPILE

10. Statements in `main()`, :

```
SimpleClass A, B ;
A.priv1 = A.priv1 + B.priv2 ;
```

COMPILES DOES NOT COMPILE

Questions 11–22 refer to the following declarations:

```
class BaseClass {
public:
   int baseData ;
} ;

class DerivedClass : public BaseClass {
public:
   int moreData ;
} ;

void someFunc(BaseClass& x) ;

BaseClass base1 ;
BaseClass *basePtr ;
DerivedClass deriv1 ;
DerivedClass *derivPtr ;
```

11. `base1 = deriv1 ;` COMPILES DOES NOT COMPILE

12. `deriv1 = base1 ;` COMPILES DOES NOT COMPILE

13. `basePtr = &deriv1 ;` COMPILES DOES NOT COMPILE

14. `derivPtr = &base1 ;` COMPILES DOES NOT COMPILE

15. `basePtr->baseData = 5 ;` COMPILES DOES NOT COMPILE

16. `basePtr->moreData = 5 ;` COMPILES DOES NOT COMPILE

17. `derivPtr->baseData = 5 ;` COMPILES DOES NOT COMPILE

18. `derivPtr->moreData = 5 ;` COMPILES DOES NOT COMPILE

19. `someFunc(base1) ;` COMPILES DOES NOT COMPILE

20. `someFunc(deriv1) ;` COMPILES DOES NOT COMPILE

21. `someFunc(&base1) ;` COMPILES DOES NOT COMPILE

22. `someFunc(&deriv1) ;` COMPILES DOES NOT COMPILE

II. Multiple Choice (2 points each)

For each question in this section, circle **ONE** answer. Choose the **BEST** answer.

1. Memory for a dynamically allocated array of `int` is stored in:
   (a) the `main()` function
   (b) the heap
   (c) the stack
   (d) all of the above

2. Suppose that function `foo()` is a friend of class `myClass`.
   (a) `foo()` can modify `const` objects of type `myClass`
   (b) `foo()` can be used to clone `myClass` objects.
   (c) `foo()` can modify private members of `myClass` objects
   (d) all of the above

3. A copy constructor is invoked when
   (a) an object is passed by reference
   (b) an object is assigned to another object
   (c) an object is passed by value
   (d) all of the above

4. When an object `X` of derived class is created,
   (a) the base class constructor is called
   (b) the derived class constructor is called
   (c) the base class constructor is called before the derived class constructor
   (d) all of the above

5. When an object `X` of derived class goes out of scope,
   (a) the bass class destructor called
   (b) the derived class destructor is called
   (c) the derived class destructor is called before the base class destructor
   (d) all of the above

6. The term "memory leak" refers to a situation where

   (a) unused dynamic memory does not get deallocated.
   (b) bits in memory are flipped by gamma radiation.
   (c) files on a hard disk gets corrupted.
   (d) `new` allocates more memory than the programmer requested.

7. If `delete` is invoked on a `NULL` pointer,

   (a) it will result in a memory leak.
   (b) a segmentation fault will occur.
   (c) the result is dependent on the system you are using.
   (d) none of the above.

8. In C++, polymorphism is achieved using

   (a) class derivation and virtual functions
   (b) interfaces and aggregation
   (c) auto pointers and macros
   (d) templates and instantiated classes

9. After executing the following code fragment:

```
int x = 4 ;
int * const ptr = &x ;
```

   (a) the value of `x` cannot be modified using `ptr`.
   (b) we cannot change the value of `ptr` and have it point to a different `int` variable.
   (c) we cannot change the value of `x` or of `ptr`.
   (d) `x` becomes a `const int` variable.

10. You should use a `const` reference parameter in a function when:

    (a) you need to return multiple values from the function.
    (b) you don't want to make a copy of the actual parameter.
    (c) you promise not to change the value of the actual parameter.
    (d) you don't want to make a copy of the actual parameter and you promise not to change its value.

11. Suppose that a program runs in $O(n)$ time. If we double the value of n, we would expect:

    (a) the running time of the program to be about the same.
    (b) the running time of the program to be roughly double.
    (c) the running time of the program to be roughly quadruple.
    (d) the running time of the program to be approximately squared.

12. Suppose that a program runs in $O(n^2)$ time. If we double the value of n, we would expect:

    (a) the running time of the program to be about the same.
    (b) the running time of the program to be roughly double.
    (c) the running time of the program to be roughly quadruple.
    (d) the running time of the program to be approximately squared.

III. Short Answers (4 points each)

1. When should a base class have a virtual destructor?

2. What is an abstract base class? How is it different from a non-abstract base class?

3. List 3 member functions that a class must have if it has dynamically allocated data.

4. After the following code fragment, what are the values of `x` and `y`?

```
int x = 4, y = 7 ;
int *ptr ;

ptr = &x ;
*ptr = x + y ;
ptr = &y ;
```

5. List one disadvantage of using templates over class derivation to achieve code reuse.

IV. Coding (6 points each)

1. Write the class definition (header file) for a `Book` class. Use static, const, virtual and & (references) whenever appropriate. The class must have these members:

   - A data member `m_title` that stores the title of the book as a C++ `string` object
   - A dynamically allocated array (not vector) of `string` named `m_authors`. Each item in `m_authors` is an author of the textbook.
   - An `int` data member `m_numAuthors` that stores the number of authors.
   - An `int` data member `m_pages` that stores the number of pages in the book.
   - a default constructor
   - a copy constructor
   - an alternate constructor that allows the client to initialize the book title and the number of pages. (Authors will be added using `addAuthor()`.)
   - a destructor
   - a function `addAuthor()` that takes a `string` parameter and adds that parameter as an author in the array of authors.
   - a function `print()` that prints out the title, authors and number of pages in the book.
   - an overloaded assignment operator.

2. Write an implementation of the `print()` function of the `Book` class as it would appear in a `.cpp` file.

3. Write an implementation of the copy constructor for the `Book` class as it would appear in a `.cpp` file. You may not use the overloaded assignment operator.

4. Write an implementation of the destructor for the `Book` class as it would appear in a `.cpp` file.

5. Write an implementation of the `addAuthor()` member function for the `Book` class as it would appear in a `.cpp` file. *Note:* keep things simple and just make a new array of authors. Also you may assume that the C++ `string` class defines an overloaded assignment operator.

6. Derive a class `TextBook` from `Book`. The `Textbook` class should add an `int` data member `m_edition` that stores the edition of the book. It should redefine the `print()` member function. Also, the alternate constructor for `Textbook` should allow the client to specify the title, number of pages and the edition of the `Textbook`.

   Write the class definition of `Textbook` as it would appear in a header file.

7. Write an implementation of the alternate constructor for `Textbook` as it would appear in a `.cpp` file. For full credit, you should use member initializers.

8. Write an implementation of the redefined `print()` function for `Textbook` as it would appear in a `.cpp` file.

9. The `print()` member function could be declared as a virtual member function in `Book`. When would you make `print()` virtual? and when would you not?