Name: _____

Username: _____

Section: (check one)
- ☐ 0101 - Justin Martineau, Tuesday 11:30am
- ☐ 0102 - Sandeep Balijepalli, Thursday 11:30am
- ☐ 0103 - Justin Martineau, Tuesday 1:00pm
- ☐ 0104 - Sandeep Balijepalli, Thursday 1:00pm
- ☐ 0105 - Don Miner, Wednesday 10:00am

	Score	Max
I.		16
II.		14
III.		32
IV. p. 8		12
p. 9		8
p. 10		12
V. p. 11		7
p. 12		14
p. 13		14
p. 14		7
p. 15		14
Total		150

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. True/False (1 point each)

Circle **ONE** answer either TRUE or FALSE. Choose the **BEST** answer.

1. Encapsulation is an OOP term used to describe the situation where one class has another class as a data member.

   TRUE FALSE

2. Aggregation is an OOP term used to describe an inheritance hierarchy where related classes are derived from the same base class.

   TRUE FALSE

3. The friend of a class is allowed to invoke its private member functions.

   TRUE FALSE

4. Memory for local variables comes from the heap.

   TRUE FALSE

5. The scope resolution operator `::` can be used to reference a variable or function in a different namespace.

   TRUE FALSE

6. There is only one copy of a static data member of a class for all the objects of that class.

   TRUE FALSE

7. A static member function is any function that has a static data member as one of its local variables.

   TRUE FALSE

8. The compiler-supplied assignment operator performs a deep copy.

   TRUE FALSE

9. A base class should have a virtual destructor whenever the class has a pure virtual function.

   TRUE FALSE

10. A virtual `clone()` function is needed to copy polymorphic objects because copy constructors are not inherited and cannot be made virtual.

    TRUE FALSE

11. In a privately derived class, the private member functions have access to the private member functions of the base class.

    TRUE FALSE

12. You cannot instantiate an object of an abstract base class.

    TRUE FALSE

13. Every object of a derived class has its own virtual function table.

    TRUE FALSE

14. A program running in $O(n \log n)$ time is much slower than one taking $O(n^2)$ time for large data sets.

    TRUE FALSE

15. It takes $O(n^2)$ time to do n insertions in a sorted array with n items.

    TRUE FALSE

16. The STL container class `set` uses a Fibonacci heap as its underlying data structure.

    TRUE FALSE

II. 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 1–3 refer to the following declarations:

```
int foo( int& m, const int& n, int t ) ;
int a, b, c ;
```

Note that the three parameters have different types.

1. `foo(a, 7, c) ;`

   COMPILES DOES NOT COMPILE

2. `foo(1, b, c) ;`

   COMPILES DOES NOT COMPILE

3. `foo(a, b, 3) ;`

   COMPILES DOES NOT COMPILE

Questions 4–7 refer to the following declarations:

```
class AClass {
public:
   AClass() ;
   int pub1, pub2 ;
   void mfunc1(AClass& B) ;


private:
   int priv1, priv2 ;
   void mfunc2(const AClass& B) ;

} ;
```

4. A statement in `mfunc1()` :

   ```
   priv1 = pub1 + pub2 ;
   ```

   COMPILES DOES NOT COMPILE

5. A statement in `mfunc1()` :

   ```
   B.priv1 = pub1 + pub2 ;
   ```

   COMPILES DOES NOT COMPILE

6. A statement in `mfunc2()` :

   ```
   B.priv1 = pub1 + pub2 ;
   ```

   COMPILES DOES NOT COMPILE

7. Statements in `main()`, :

   ```
   AClass A, B ;
   B.priv1 = A.pub1 + A.pub2 ;
   ```

   COMPILES DOES NOT COMPILE

Suppose that BBB and DDD are classes and that DDD is publicly derived from BBB. Questions 8–14 refer to the following declarations:

```
void gee1(BBB  x) ;
void gee2(BBB& x) ;
void gee3(BBB* x) ;
void gee4(DDD* x) ;

BBB X, *Xptr ;
DDD Y, *Yptr ;
```

8. `gee1(Y) ;`

   COMPILES DOES NOT COMPILE

9. `gee2(Y) ;`

   COMPILES DOES NOT COMPILE

10. `gee2(&Y) ;`

    COMPILES DOES NOT COMPILE

11. `gee3(&Y) ;`

    COMPILES DOES NOT COMPILE

12. `gee4(&X) ;`

    COMPILES DOES NOT COMPILE

13. `Xptr = &Y ;`

    COMPILES DOES NOT COMPILE

14. `Yptr = &X ;`

    COMPILES DOES NOT COMPILE

III. Multiple Choice (2 points each)

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

1. A copy constructor is invoked when
   (a) an object is passed by value
   (b) an object is returned by value
   (c) an object is initialized by another object of the same class
   (d) all of the above

2. In the implementation of an overloaded assignment operator, we must check for self-assignment because:
   (a) self-assignments will be rejected by the compiler
   (b) self-assignments are prohibited by the coding standards
   (c) if the object has dynamically allocated members, we do not want to deallocate memory twice.
   (d) if the object has dynamically allocated members, self-assignment will be an infinite loop.

3. The following `catch` block will catch what types of exceptions?

```
catch (MyExceptionClass& e) {
   ...
}
```

   (a) exceptions of class `MyExceptionClass` and of any class derived from `MyExceptionClass`.
   (b) exceptions of class `MyExceptionClass` and of any class that `MyExceptionClass` is directly or indirectly derived from.
   (c) only exceptions of class `MyExceptionClass`.
   (d) only pointers to objects of class `MyExceptionClass`.

4. You should never throw an exception in a
   (a) constructor
   (b) destructor
   (c) the `main()` function
   (d) private member functions

5. When an object `X` of derived class goes out of scope,
   (a) only the derived class destructor is called.
   (b) the base class destructor is automatically called, followed by a call to the derived class destructor.
   (c) the base class destructor is automatically called after the derived class destructor has finished.
   (d) none of the above

6. When an object `X` of derived class is created,

   (a) only the derived class constructor is invoked.

   (b) the base class constructor is automatically called, followed by a call to the derived class constructor.

   (c) the base class constructor is automatically called after the derived class constructor has finished.

   (d) none of the above

7. To deallocate a dynamically allocated array `A` of 20 `int`, we use:

   (a) `delete A ;`

   (b) `delete A[] ;`

   (c) `delete A[20] ;`

   (d) `delete [] A ;`

8. If `delete` is invoked with a NULL pointer,

   (a) nothing happens

   (b) a segmentation fault would occur

   (c) a bus error would occur

   (d) a memory leak results

9. The advantage of using a sorted linked list over an unsorted link list is:

   (a) insertion times are much faster

   (b) deletion times are much faster

   (c) binary search can be used to speed up search times

   (d) none of the above

10. In a private derivation,

    (a) Only the private members of the base class are inherited by the derived class.

    (b) Only the public members of the base class are inherited by the derived class.

    (c) The public members of the base class become private members of the derived class.

    (d) The private members of the base class become public members of the derived class.

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

    (a) the computer systems RAM becomes corrupted over time.

    (b) unused dynamic memory does not get deallocated.

    (c) private data members become visible to non-member functions.

    (d) private data members in a base class become visible to member functions of a derived class.

12. In C++, polymorphism is achieved using

    (a) macros and `void *` pointers
    (b) `void *` pointers and dynamic casting
    (c) derived classes and virtual functions
    (d) `mutable` data members

13. When an exception is thrown, control of the program is given to

    (a) the most recently called function that has a catch block that matches the type of the object being thrown.
    (b) the least recently called function that has a catch block that matches the type of the object being thrown.
    (c) the function that called the function that threw the exception.
    (d) the `main()` function.

14. The advantage of using an `auto_ptr` is that you can

    (a) allocate memory without using `new`
    (b) deallocate memory without using `delete`
    (c) have a pointer that automatically assumes the type of its "pointee".
    (d) have a "raw address" pointer that has no type.

15. Macros in C

    (a) are defined using `#define`.
    (b) let you write in-line functions.
    (c) do not understand type.
    (d) all of the above

16. The keyword `const` in the declaration:

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

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

IV. Short Answers (4 points each)

1. Explain when it is appropriate to use a `const` reference parameter in a function.

2. List 3 properties of a function that are parts of its function signature and 1 property that is not.

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`, `y` and `z`?

```
int x = 111, y = 999 ;
int &z = y ;

z = x + 5 ;
z = y - x ;
z = x ;
```

5. The following program exploits the function overloading feature of C++. Write down the output of the program.

```
#include <iostream>
using namespace std ;

void PrintIt (char c) {
   cout << "char c = " << c << endl ;
}

void PrintIt (int n, float z) {
   cout << "int n = " << n << "  float z = " << z << endl ;
}

void PrintIt (float x, float y) {
   cout << "float x = " << x << "  float y = " << y << endl ;
}

int main() {
   char d = 'x' ;
   int s = 2, t = 3 ;
   float v = 4.1, w = 5.9 ;

   PrintIt(d) ;
   PrintIt(s, w) ;
   PrintIt(v, w) ;
   PrintIt(s, t) ;
   PrintIt(v, t) ;
}
```

6. List two advantages of using templates over class derivation to achieve code reuse.

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

8. List two advantages of using the STL `set` container class over the STL `vector` class.

V. Coding (7 points each)

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

   - the name of the owner stored as a string
   - a default constructor
   - an alternate constructor that allows the client to initialize the bag
   - a copy constructor
   - a `GetSize()` function that returns the number of marbles in the bag.
   - a vector that holds the diameter of each individual marble in the bag.
   - an `AddMarble()` function that adds a marble of specified diameter to the bag.
   - an overloaded `+=` operator that allows you to add all of the contents of another `BagOfMarbles` into this one.

2. Give the implementation of the `BagOfMarbles` copy constructor. This copy constructor should perform a *deep copy.*

3. Give the implementation of the overloaded `+=` operator. Note that once the contents of the second `BagOfMarbles` is given to the first `BagOfMarbles`, the second `BagOfMarbles` becomes empty.

4. Write a templated function `TheMax()` that takes an array of any type of items and returns the index of the item that is largest according to the $<$ operator.

5. Finish the implementation of the `PrintList()` function below. Use the iterator `it` to step through the list of strings `L` and print out each string in the list.

```
#include <list>
#include <string>
using namespace std ;

void PrintList (list<string> L) {

    list<string>::iterator it ;
```

6. Consider the simple `Node` class defined below that might be used in a linked list:

```
class Node {
public:
   Node() ;            // default constructor
   Node(int data) ;    // alt. constructor, initializes m_data
   virtual ~Node() ;   // virtual destructor, just in case

   Node* next ;
   void Set(int data) ;     // stores in m_data
   int Get() ;              // retrieves m_data
   virtual void Print() ;   // prints out m_data

private:
   int m_data ;
} ;
```

Write the definition of a class `AnimalNode` that is a public derivation of `Node`:

- `AnimalNode` adds a string data member `name` and a `float` data member `weight`.
- `AnimalNode` will use `m_data` to store the age of the animal.
- `AnimalNode` should override the `Print()` member function.
- `AnimalNode` should have an alternate constructor that allows the client to initialize `name`, `weight` and `m_data`.

7. Implement the alternate constructor for `AnimalNode`. You may assume that the member functions for the base class `Node` have been correctly implemented.

8. Rewrite the definition of the `Node` class, so it is a templated class. The type of `m_data` must be the template's type parameter. The type of the member functions should be adjusted accordingly.