Name: ______________________________

Username: ______________________________

Section: (check one)

□ 07 - Sushant Athley, Tuesday 11:30am

□ 08 - Aishwarya Bhide, Thursday 11:30am

□ 09 - Phanindra Kumar, Tuesday 2:30pm

□ 10 - Phanindra Kumar, Thursday 2:30pm

□ 16 - Sudip Mittal, Tuesday 10:00am

□ 17 - Sushant Athley, Thursday 4:00pm

	Score	Max
I.		10
II. p. 2		12
p. 3		12
p. 4		16
III. p. 5		10
p. 6		20
p. 8		10
p. 9		10
Total		100

Instructions:

1. This is a closed-book, closed-notes exam.
2. You have 75 minutes for the exam.
3. Calculators, cell phones and laptops must be put away.
4. Clearly indicate your final answer.

I. Multiple Choice (2 points each)

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

1. C++ objects can be

   (a) the return value of a function.

   (b) passed by value to a function.

   (c) passed by reference to a function.

   [✓ (d) all of the above.]

2. Suppose that `A` and `B` are `AClass` objects and `change()` is an `AClass` member function. Assume that `AClass` does not contain any const data members. During the execution of:

   ```
   A.change(B) ;
   ```

   (a) `change()` can change all data members of `A` but none of the data members of `B`.

   (b) `change()` can change only the public data members of `A` and `B`.

   [✓ (c) `change()` can change all data members of `A` and all data members of `B`.]

   (d) none of the above.

3. Suppose that `A` and `B` are `AClass` objects and `munge()` is not a member function of `AClass` and is not a friend of `AClass`. Assume that `AClass` does not contain any const data members. During the execution of:

   ```
   munge(A,B) ;
   ```

   (a) `munge()` can change all data members of `A` but none of the data members of `B`.

   [✓ (b) `munge()` can change only the public data members of `A` and `B`.]

   (c) `munge()` can change all data members of `A` and all data members of `B`.

   (d) none of the above.

4. Suppose that `alter()` is a **const** member function of `AClass` with function prototype:

   ```
   int alter(AClass& B) const ;
   ```

   Assume that `AClass` does not contain any const data members. Let `A` and `B` be `AClass` objects. Then, during the execution of

   ```
   i = A.alter(B) ;
   ```

   [✓ (a) `alter()` can change all data members of `B` but none of the data members of `A`.]

   (b) `alter()` cannot return the value of private data members of `A`.

   (c) `alter()` can change the public data members of `A` but not the private data members of `A`.

   (d) `alter()` can change the public data members of `B` but not the private data members of `B`.

5. Suppose that there are exactly 2 functions named `Duo` in a program and those functions have the following prototype:

```
void Duo(int a, int b) ;     // first Duo
void Duo(int a, float b) ;   // second Duo
```

Then, the following function call would:

```
int i = 3 ;
float x = 4.0 ;

Duo(i, x) ;
```

(a) call the first `Duo`.

[✓ (b) call the second `Duo`.]

(c) result in a compile time error.

(d) result in a run-time error.

II. Short Answers (4 points each)

1. Give two *different* examples where a copy constructor is invoked.

2. Explain the difference between shallow copy and deep copy.

3. Explain the difference between a copy constructor and an overloaded assignment operator.

4. Give two *different* examples where a destructor is invoked.

5. Write the code that declares a pointer named `ptr` that points to `float`. Then write the code that assigns a dynamically allocated array of 17 `float`s to `ptr`.

6. Suppose that `AClass` is a class. Write the code to declare a pointer named `ptr` that points to `AClass` objects. Then write the code that assigns the address of a dynamically allocated `AClass` object to `ptr`.

7. Why do some classes need destructors?

8. Describe one difference between a static data member in a class and a non-static data member in a class.

9. What is the value of the `this` pointer during the execution of the body of a member function?

10. What is the advantage of having a dummy header in a linked list?

III. Coding (10 points each)

Length. *Questions 1-3 refer to the following class definition:*

```
// Stores the length measured in feet, inches and sixteenths of an inch.
class Length {

public:
   Length() ;
   Length(unsigned int feet, unsigned int inches, unsigned int sixteenth) ;
   void timestwo() ;      // doubles the length
   void print() const ;   // pretty print

   unsigned int GetFeet() const ;      // accessors
   unsigned int GetInches() const ;
   unsigned int GetSixteenth() const ;

private:
   void adjust() ;         // tidy up units
   unsigned int m_feet ;   // data members
   unsigned int m_inches ;
   unsigned int m_sixteenth ;
} ;
```

Consistency Requirement: All member functions of the Length class must maintain this consistency requirement: there must never be more than 11 inches in m_inches or more than 15 sixteenths of an inch in m_sixteenth.

1. Implement the private member function adjust() which checks whether the consistency requirement defined above has been violated. If it has, adjust() should change the values of the data members to satisfy the consistency requirement and preserve the length represented.

   Write out the entire function as it would appear in the Length.cpp implementation file.

2. Implement the alternate constructor for the `Length` class using member initializers. Remember the consistency requirement.

   Write out the entire function as it would appear in the `Length.cpp` implementation file.

3. Implement the member function `timestwo()` for the `Length` class. The `timestwo()` function doubles the length stored in the object. Remember the consistency requirement.

   Write out the entire function as it would appear in the `Length.cpp` implementation file.

Linked Lists. The following are the class definitions for a singly linked list much like the one used in Project 3. The linked list uses a dummy header at the beginning of the list.

Questions 4 & 5 refer to these class definitions:

```
// The node used in List
class Node {
public:
   Node(int data);
   int m_data;
   Node* next;
};


// List is a linked list of ints
class List {
public:
   // Creates a default empty list
   List();

   // Creates a copy of another list
   List(const List &rhs);

   // Destructor
   ~List();

   // Assignment operator
   const List& operator=(const List &rhs);

   // Insert "data" into the list
   void insert(int data);

   // Returns the size of the list
   unsigned int size() const;

   // Returns a pointer to an array with the same data
   int *ConvertToArray() const ;

   // Compute the sum of all the nodes
   int Sum() const ;

   // Remove the last node in the linked list, if it exists
   void RemoveLast() ;

private:
   Node* m_head;

};
```

4. The `Sum()` member function computes the sum of the `m_data` stored in each node of the linked list (not counting the dummy header). If the list is empty, `Sum()` returns 0.

   Write an implementation of `Sum()` as it would appear in a `.cpp` file.

5. The `ConvertToArray()` member function dynamically allocates an array of `int` then copies the nodes of the linked list (excluding the dummy header) to the array. A pointer to the dynamically allocated array is returned by `ConvertToArray()`.

   Write an implementation of `ConvertToArray()` as it would appear in a `.cpp` file. You may assume that the `size()` member function has been implemented properly.