ECE2036 **Fall Semester, 2012**

Final Exam

December 10, 2012 *NAME:*____

OPEN NOTES, CALCULATORS ALLOWED, NO OLD TESTS/EXAMS.

1. Constructors and Destructors (**15 Points**)

   In the program `Constructors-Destructors.cc` attached, identify where (what line number) each of the default constuctors, int constructors, copy constructors, and destructors are called for each class `A` and `B`. Specify which line of code causes each of the above and a brief explanation of why the constructor was called. Be sure to note that the addition operator for classes `A` and `B` are both defined differently and implemented differently. As an example of how to fill in the table, one entry for the `A` int constructor is filled in.

`A` **Default Constructor**	**Line Number**	**Explanation**
`A` **int Constructor**	**Line Number**	**Explanation**
	48	Declaration of local variable “a” with int argument
`A` **Copy Constructor**	**Line Number**	**Explanation**
`A` **Destructor**	**Line Number**	**Explanation**
`B` **Default Constructor**	**Line Number**	**Explanation**
`B` **int Constructor**	**Line Number**	**Explanation**
`B` **Copy Constructor**	**Line Number**	**Explanation**
`B` **Destructor**	**Line Number**	**Explanation**

2. Inheritance (**15 Points**)

What is printed by the attached `inheritance.cc` program? Hint. There are 10 “Hello from” messages.

(a) Hello from

(b) Hello from

(c) Hello from

(d) Hello from

(e) Hello from

(f) Hello from

(g) Hello from

(h) Hello from

(i) Hello from

(j) Hello from

3. Matrix calculator assignment (**15 Points**)

   A very buggy version of the Matrix Calculator implementation is given on the attached pages. It compiles correctly, but has at least four major flaws. Point out four mistakes in the Matrix class and explain why each mistake is a bug and what the fix is. Refer to line numbers in the handout as needed. *Hint; Look for both memory management errors and logic errors*

   (a) Mistake 1

   (b) Mistake 2

   (c) Mistake 3

   (d) Mistake 4

4. Template Linked List (**15 Points**)

   The templated linked list handout is attached.

   (a) Explain why we did not implement an indexing operator.

   (b) Why does the ListIterator postfix increment operator need a temporary variable `tmp` (see line 32) but the prefix increment does not?

   (c) Explain why we did not (and in fact could not) implement a prefix or postfix decrement operator.

   (d) The code below is intended to iterate over an existing linked list called `List`, but it has a fatal flaw and in fact won't compile. Explain what the problem is and what should change to fix it.

```
for (ListIterator it = List.Begin(); it < List.End(); ++it)
  {  // internal of the loop omitted for brevity
  }
```

   (e) Why did we not implement an `InsertBefore` function to add a new node before an existing node?

5. Templated Subroutines **10 Points**

   The code snippet on the attached page implements a `Sort` subroutine, that sorts a collection of objects specified by a pair of parameters specifying the first and (last + 1) elements to be sorted. Note that this subroutine is *generic* in the sense that we implemented it for any arbitrary type `T`. **Hint. Pay particular attention to line 14**.

   At lines 52 through 55 we attempt to instantiate the `Sort` routine with four different parameter types. Which of the four instantiations of `Sort` will compile and which will not? State reasons why you think the call will compile properly or not.

6. STL Containers **20 Points**

(a) What is printed at lines 21-22 in the attached `Containers.cc` program?

v1 size _____________ v2 size _____________ v1 front() _____________ v2.back() _____________

(b) What is printed at lines 32-33 in the attached `Containers.cc` program?

d1 front() _____________ d1.back() _____________ d1[5] _____________

(c) What is printed at lines 43-47 in the attached `Containers.cc` program?

m1.size() _____________ begin.first _____________ begin.second _____________

- -(m1.end).first _____________ - -(m1.end).second _____________

7. Static Member Functions **10 points**

   In the `Static.cc` program, we implement a class called `MyClass` that as some member variables, and two member functions `Func1()` and `Func2()`. As implemented in the `Static.cc` program both `Func1()` and `Func2()` are member functions, not *static* member functions. In fact, one of the two *should* be *static* and one of the two *cannot* be static. Fill in the answers below.

   Member function _____________ should likely be declared *static* because:

   Member function _____________ *cannot* be declared *static* because: