CMSC 202 Final

May 19, 2005

Name: ____________________________ UserID: _________________
(Circle your section)

Section:

101 – Tuesday 11:30
103 – Tuesday 12:30
105 – Tuesday 1:30
102 – Thursday 11:30
104 – Thursday 12:30
106 – Thursday 1:30

Directions

• This is a closed-book, closed-note, closed-neighbor exam.
• Read through the entire test before you begin.
• Start with the questions that are easiest for you, come back to the rest.
• Write CLEARLY, if I cannot read your writing, you will receive a zero for the problem in question.
• Feel free to continue your answer on the backs of the pages, but make sure that you indicate where your answer continues.
• When you are done, read over your answers and then bring your exam to the front of the room.
• You will need your Picture ID to hand in your exam.

Score

<table>
<thead>
<tr>
<th>Page Number</th>
<th>Points Possible</th>
<th>Points Earned</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Have a Great Summer!
Section 1: True/False (10 pts total, 1 pt each)
Read each statement carefully and write true or false on the blank to the left.

_________ 1. An abstract class has one or more virtual functions.

_________ 2. A class that is intended to be a base class need not have a virtual destructor.

_________ 3. A derived class has direct access to the base class’s private data members

_________ 4. An inline function guarantees that a compiler will replace the function call with the body of the function.

_________ 5. When using exceptions, only one try block is allowed per program while many catch blocks are allowed.

_________ 6. The following is a valid collection of elements of a Set<int> object from the STL: {2, 4, 3, 4, 6, 3, 2, 1, 5}

_________ 7. Templated classes and functions are generated and bound at compile-time.

_________ 8. Assume there is a class named B that inherits privately from A, and a class C that inherits publicly from B, the following object instantiation is acceptable:
   C* ptr = new A;

_________ 9. Static binding describes the ability of the compiler to bind an object to the correct method in an inheritance hierarchy.

_________ 10. Assume there is a class named B that inherits publicly from A, where A is an abstract class, the following object instantiations are acceptable:
    A* a = new A();
    B* b = new B();
    A* c = new B();
**Short Answer**

Complete each of the short-answer coding questions. You may assume that the questions build on each other and that previously implemented lines can be used in later questions.

Assume there is a class named Ball with derived classes named BeachBall, FootBall, and VolleyBall.

11. (4 pts) Define a vector of Ball pointers.
   Define an iterator to this vector.

12. (2 pts) Assume there are already 4 Balls (of various subtypes) in the vector.
    Add a BeachBall to the vector.

13. (4 pts) Assume that the insertion operator is overloaded for all Ball types.
    Using a for-loop and the iterator, iterate through the vector, printing each ball to the screen.

14. (10 pts) Assume that the < (less than) operator is defined for all Ball types and returns a boolean (true=current object is less). Define a templated function that finds the Smallest item in the vector and returns its index.
15. (15 pts) Assume that the following operators are defined for all STL iterators: <, >, <=, >=, ==, !==, ++, and --. Using one or more of these operators, overload the subtraction operator for the List iterator that will subtract one iterator from another, returning an integer representing the number of items between them. Do not overload the operator as a class method.
Ex: If there are 5 items in a vector, then vec.end() – vec.begin() should return 5.

16. (5 pts) Assume the BeachBall has an overloaded constructor that accepts a radius. Assume there is also a related mutator. Assume the following lines are defined:

```cpp
BeachBall a(7.0);
BeachBall b(6.0);
const BeachBall c(5.0);
const BeachBall* p = &a;
BeachBall* const q = &b;
```

Identify whether the following lines are illegal, if so, describe why.

```cpp
p->SetRadius(1.0);
```

```cpp
q->SetRadius(2.0);
```

```cpp
p = &c;
```

```cpp
q = &c;
```

```cpp
p->SetRadius(1.0);
```

```cpp
q->SetRadius(2.0);
```
17. (10 pts) Assume that the BeachBall constructor used in the previous question throws a NegativeException and an unknown exception. Write the try/catch block that will create a BeachBall and correctly catch the exceptions. Assume there is a message() method that returns the exception’s message if necessary.

18. (10 pts) Assume the BeachBall has a dynamic member of type pointer to string called m_name and a double called m_radius. Assume the constructor accepts values for both as parameters. Implement the constructor that throws a NegativeException if the radius parameter is less than zero.

19. (5 pts) Implement the destructor for the BeachBall class, assume it has been prototyped in the class definition.

20. (5 pts) Free all the memory used by the vector.
Class Implementations

21. (15 pts) Write the class definition (header file) for the Ball class. Use constants, virtual and references whenever appropriate. The Ball class has the following members:
   a. radius data member, double – inherited classes should have access
   b. Default constructor, sets radius to zero
   c. Non-default constructor, sets radius to parameter value if valid
   d. Copy constructor
   e. Destructor – destroys object
   f. Inflate method – increases radius by 0.1
   g. Deflate method – decreases radius by 0.1, if possible
   h. Volume method – possibly overridden by inherited classes, calculates and returns volume of Ball
   i. Print method – must be overridden by inherited classes
22. (15 pts) Write the **class definition** (header file) for the BeachBall class. Use constants, virtual and references whenever appropriate. The BeachBall class has the following members:

a. (10 pts) **BeachBall**, inherits from **Ball**
   i. **name** data member, pointer to a string
   ii. **Default** constructor
   iii. **Non-default** constructor, uses non-default constructor of Ball
   iv. **Copy** constructor, uses copy constructor of Ball
   v. **Destructor** – destroys any dynamic memory
   vi. **Print** method – overrides Ball’s version

b. (5 pts) Implement two versions of the **Copy** constructor (shallow and deep)
   i. **Shallow Copy**
   ii. **Deep Copy**
23. (10 pts) Write the definition for a templated collection class called Bin. Use constant and reference as appropriate. Bins have the following members:
   a. **items** data member – dynamic structure to hold collection of items
   b. **Default** Constructor
   c. **addItem** method – add item (parameter) at “end” of collection
   d. **removeItem** method – removes item at “beginning” of collection, returns that item
   e. **overloaded [] operator** (array access) – accepts an integer i, returns the ith item in the collection

24. (5 pts) Implement **addItem**

25. (5 pts) Implement **removeItem**

26. (5 pts) Implement the **overloaded [] operator**
27. (5 pts) Why do we want to allow derived classes to override the Volume method of the Ball class? Provide an example to support your answer.

28. (5 pts) What method is missing from both the Ball and BeachBall class? Why must we include this method?

29. (5 pts) Briefly describe the process of "stack unwinding" which occurs if an exception is not caught.

30. (10 pts) Briefly describe the generally accepted classifications of exception safety that a function can make?
Extra Credit

1. (5 pts) Assume that you want to implement a templated Queue (FIFO – first in, first out), but only have access to a Stack (LIFO – last in, first out) with the following methods:
   - push, pushes an item onto the stack.
   - pop, pops an item from the stack but does not give this item to the programmer to use.
   - top, gives the programmer a reference to the top of stack item; no change is made to the stack.
   - empty, the usual boolean function (true = stack is empty).
   - size, reports the number of items on the stack.

Describe briefly how you would use a Stack to implement a Queue.
Draw a picture to clarify your strategy.
You may assume that your Queue class supports the following:
   - private data member: `Stack<T> stack;`
   - insert method – insert the parameter into the “end” of the Queue
   - remove method – remove the “first” item from the Queue and return that item

2. (10 pts) Implement the insert() and remove() methods for your Queue using the STL stack. You may allocate any additional memory necessary, but cannot use any other data structure other than Stack.
Extra Credit – Part Deux

3. (5 pts) Define a function object called Mystery that will accept an integer by reference, multiply it by 2 and then add the value of the private data member (set via the constructor). Declare all methods inline.

4. (5 pts) Use this Mystery function with the for_each function on a vector of integers (you can assume the code below). Use 7 as the parameter to the Mystery constructor.

```cpp
vector<int> vec;
vec.push_back(1);
vec.push_back(2);
// ... other values are added...
```

5. (2 pts) If you could have any ONE superpower (fly, x-ray vision, super strength, speed, time-travel, etc.), what power would you have and why? Creative answers will get credit.