CMSC 202 Final

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.
• Show your Picture ID AND Exam paper to a TA/Instructor, place in correct pile.

Score

<table>
<thead>
<tr>
<th>Page Number</th>
<th>Points Possible</th>
<th>Points Earned</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>11 (EC)</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>12 (EC)</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100 (+15 EC)</td>
<td></td>
</tr>
</tbody>
</table>

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

__________ 1. It is legal to instantiate an object of an abstract class.

__________ 2. The following lines of code correctly modify the value of 'a' to be 7.
    int a = 2;
    int* ptr = new int(a);
    ptr = 7;

__________ 3. Dynamic objects are allocated on the stack while static object are allocated on the heap.

__________ 4. The following code correctly creates and deletes an array.
    int* array = new int[ 10 ];
    delete array[ ];

__________ 5. If try/catch blocks are nested, the exception is always thrown to the outermost catch block.

__________ 6. The default overloaded operator= (provided by the compiler) results in a shallow copy of memory.

__________ 7. When using dynamic memory, one should always overload the copy-constructor and should be sure to protect from self-assignment (i.e. assigning A to itself).

__________ 8. Derived classes can use, modify or extend methods from their parent class(es).

__________ 9. When polymorphism is used in C++, the base-class destructor is called before the derived-class destructor.

__________ 10. Assume that myVector is a vector of integers, myVector.end() returns an iterator that points to the last item in myVector.
**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 Rider with derived classes named Skier and Snowboarder.

11. (1 pt) Define a **dynamic array** of **Rider pointers**. Assume that the size of the array is in a variable named 'size'.

12. (1 pt) Assume there are already 4 Riders (of various subtypes) in the vector. Add a **Snowboarder** to the array.

13. (3 pts) Assume that the **insertion** operator is **overloaded** for all **Rider** types. Using a **for-loop**, iterate through the array, **printing** each rider to the screen.

14. (7 pts) Assume that the **>** (**greater than**) operator is defined for all Rider types and returns a **boolean** (\(a > b \equiv \text{true if } a \text{ is a better Rider than } b\)). Define a **templated** function (the function should now know what a Rider is) that finds the **Best** item in the array and **returns the object**.
15. (6 pts) Assume the Skier has an overloaded constructor that accepts a skill-level (1 -> 9, beginner -> advanced). Assume there are also a related mutator and an accessor. Assume the following lines are defined:

```cpp
Skier a(6);
const Skier b(1);
```

Identify whether the following lines are compilable. If not, describe why. Assume each chunk of code is examined in isolation of the others.

<table>
<thead>
<tr>
<th>Will Compile</th>
<th>Code…</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Yes/No)?</td>
<td></td>
</tr>
<tr>
<td>_____________</td>
<td>const Skier* p = &amp;a;</td>
</tr>
<tr>
<td></td>
<td>p-&gt;SetSkillLevel(8);</td>
</tr>
<tr>
<td>_____________</td>
<td>Skier* const q = &amp;a;</td>
</tr>
<tr>
<td></td>
<td>q-&gt;SetSkillLevel(8);</td>
</tr>
<tr>
<td>_____________</td>
<td>const Skier* r = &amp;b;</td>
</tr>
<tr>
<td></td>
<td>r-&gt;SetSkillLevel(8);</td>
</tr>
<tr>
<td>_____________</td>
<td>Skier* const m = &amp;b;</td>
</tr>
<tr>
<td></td>
<td>m-&gt;SetSkillLevel(2);</td>
</tr>
<tr>
<td>_____________</td>
<td>const Skier* p = &amp;a;</td>
</tr>
<tr>
<td></td>
<td>p = &amp;b;</td>
</tr>
<tr>
<td>_____________</td>
<td>Skier* const q = &amp;a;</td>
</tr>
<tr>
<td></td>
<td>q = &amp;b;</td>
</tr>
</tbody>
</table>

16. (1 pts) What limitation must be placed on the skill-level accessor of the Skier classes to have the following code compile?

```cpp
const Skier * t = &b;
b.GetSkillLevel();
```
17. (10 pts) Assume that the Skier constructor used in the previous question throws an OutOfRange and some other exception. Write a loop that will create 10 Skiers and put them into the original dynamic array (assume it is empty), using 5 to 15 (consecutively) as the skill-level parameter. Using a try/catch block, correctly catch the exceptions. If an OutOfRange exception is caught, the default constructor for the Skier should be used and processing should continue with the next skier. If some other exception is caught, the exception should be re-thrown.

18. (5 pts) Implement the Skier constructor that accepts a single integer parameter (skillLevel). Assume there is a data member named 'm_skillLevel'. If the skillLevel parameter is less than 1 or greater than 9, throw an OutOfRange exception. Ignore the other exception described in the previous question.
Class Implementations

19. (10 pts) Write the class definition (header file) for the Rider class. Use static, constants, virtual and references whenever appropriate. The Rider class has the following members:
   a. **skillLevel** data member, integer, min of 1, max of 9
   b. **MaxSkillLevel** data member, integer, represents the maximum skill level
   c. **MinSkillLevel** data member, integer, represents the minimum skill level
   d. **Default** constructor, sets skillLevel to minimum
   e. **Non-default** constructor, sets skillLevel to parameter value if valid
   f. **Copy** constructor – copies parameter
   g. **Destructor** – destroys object
   h. **GetSkillLevel** – returns the Rider's skill level
   i. **SetSkillLevel** – sets the Rider's skill level
   j. **ReplaceBindings** – method to be overridden by derived classes
20. (10 pts) Write the **class definition** (header file) for the **Skier** class. Use static, constants, virtual and references whenever appropriate. Assume there is a **Ski** class that represents a **single ski**. The **Skier** class has the following members:

a. **Skier**, inherits from **Rider**
   
i. **leftSki** dynamic data member that is the left ski
   
ii. **rightSki** dynamic data member that is the right ski
   
iii. **Default** constructor, a skier initially has **no skis**
   
iv. **Copy** constructor
   
v. **Destructor** – destroys any dynamic memory
   
vi. **ReplaceBindings** – replaces the bindings on both skis

21. (2 pts) Discuss the **difference** between a **shallow** and **deep** copy for the **copy**-constructor of the Skier class. Use an **example** to **illustrate** (no code).
22. (3 pts) **Implement** the **copy** constructor of the **Skier** class using a **deep** copy.

23. (3 pts) Assume that a skier can have an **entire collection** of Skis, but that he must **choose only 2** from that collection to **use that day** (his "**Current Skis**"). Assuming that we represent the **set of skis** as a **vector** of **pointers** to Skis, briefly describe **two ways** to represent his **chosen pair**. **Compare** and **contrast** these two strategies, discussing the **time, space, and access tradeoffs** between them.

24. (2 pts) **Implement** the **destructor** for the **Skier** class.
(3 pts) Assume that we would like to add this collection of foot-equipment (i.e. Skis or Snowboards) to the base class, Rider. Prototype (i.e. forward-declare) the Rider class as a class templated on a single type of equipment.

25. (2 pts) Define the collection data member of the Rider class using a vector of pointers to the equipment-type.

26. (2 pt) Should the collection be private, protected, or public? Why?

27. (3 pts) Since the only difference between a Skier and a Snowboarder is that the Skier has 2 pieces of equipment and the Snowboarder has 1, how could you combine these two classes using templates into only the Rider class? How would you eliminate the need to store two data members (a left and right ski) for the Skier?
Exposition

28. (4 pts) **Describe** the **differences** between method **overriding** and method **overloading**. Provide an **example** to **support** your comparison.

29. (4 pts) What is the **purpose** of including a **Clone()** method in **inheritance**? Why should this **method** use **deep copies**?

30. (4 pts) Briefly **discuss** the **pros** and **cons** of using **inline** functions.

31. (4 pts) **Why** is it **important** to **protect** an object from **self-assignment** (i.e. assigning A to itself)? (Hint: think about **dynamic memory**)
Extra Credit

32. (3 pts) Assume that you want to implement a templated List (push_front, push_back, pop_front, pop_back), but only have access to a Vector with the following methods:
   • insert(iter), inserts an item before the position pointed to by the iterator parameter
   • erase(iter), removes the object pointed to by the iterator from the vector
   • Assume that the methods begin(), end(), and size() work exactly as in the STL vector class, you may also assume that the ++ and -- operators work with these iterators.

Describe briefly how you would use the above Vector to implement a List. You may assume that your List class must support the following:
   • private data member: Vector<T> list;
   • pop_back – removes last item in the List
   • push_back – insert the parameter after the last item in the List
   • pop_front – removes first item in the List
   • push_front – insert the parameter before the first item in the List

33. (3 pts) Implement the pop_back() method for your List using the Vector described above. You may allocate any additional memory necessary.
Extra Credit – Part Deux

34. (3 pts) If I had asked Extra Credit #1 in the exact opposite way (i.e. build a Vector on a Linked-List), what would be the greatest difficulty with implementing an at(i) method that returns the object in the ith position?

35. (4 pts) Write the pseudocode (or code) to implement the at(i) method of the Vector class on a private data member that is a Linked-List named 'list'. You may only use the following methods: push_front, push_back, pop_front, pop_back. You may assume that the pop_* methods return the object they have removed.

36. (2 pts) If you knew the world was going to end tomorrow, whom (if anyone) would you tell and why?