CMSC 202 Midterm October 20, 2005

Name: ________________________________ **Email ID:**______________

(Circle your section)

Section: **101** – Tuesday 11:30 **102** – Thursday 11:30

103 – Tuesday 12:30 **104** – Thursday 12:30

105 – Tuesday 1:30 **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. If you have time at the end, come back to the more challenging ones.
- 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

Page Number	Points Possible	Points Earned
2	10	
3	20	
4	15	
5	15	
6	15	
7	15	
8	10	
EC	**15**	
TOTAL	**100**	

True/False (10 pts, 1 pts each)

Decide if the following are true (T) or false (F), put the appropriate letter in the blank.

____________ 1. The `ofstream` library’s open function accepts only C-style strings.

____________ 2. The following code prints "`the`" to the screen (without quotes).

```
string message = "Quoth the Raven";
cout << message.substr(6, 8) << endl;
```

____________ 3. The following code initializes a vector of 100 integers, each having the value 0 (zero).

```
vector<int> treatsAtHouse(100, 0);
```

____________ 4. The following code removes all the items in a vector.

```
treatsAtHouse.empty();
```

____________ 5. `setwidth`(), `precision`() and `fill`() are three functions that can be used with ostreams, for example:

```
cout << setwidth(6) << precision(2)
     << fill('_') << 10.31 << endl;
```

____________ 6. The following is an example of a good precondition:

```
// PreCondition:
// passes the nbrTreats and nbrTricks
void TreatRatio( double nbrTreats,
                 double nbrTricks )
```

____________ 7. When discussing parameter passing with functions, the **parameter** refers to the variable or expression passed to the function when the function is called and the **argument** refers to the variable defined in the function definition.

____________ 8. In a class, the following function snippet implies that we cannot change the current object:

```
const bool House::RingDoorbell()
{
     // some code
}
```

____________ 9. Encapsulation refers to "using one class as a data member of another class".

____________ 10. Static methods of a class can access all private data members of that class.

Short Answer

The following questions are all related and deal with the same system. Assume that the proper header files have been included.

11. (2 pts) **Open** a **file** named "costumes.txt" for **input**.

12. (2 pts) Declare a **vector** of **strings** to store the costume information.

13. (6 pts) Use a **loop** to **read** costume data from the **file**. A **costume** consists of **one** or **more words** on the **same line**.

14. (10 pts) **Overload** a **Print**() function. One version will print a **single costume** with "**`Costume <i>: <costume description>`**" (where **`<i>`** is the costume index in the vector and **`<costume description>`** is the actual description **read** from the file. The second version will **print all** of the **costumes** in order. You may use `cout` as your ostream.

Class Construction

The following questions all have to do with the same system. Make appropriate decisions about data types, return types, const, and parameter passing. Ignore header-file guarding and includes.

15. (15 pts) Design a class called Treat. Each treat has a name, a house number where it was received, and a safety factor. House numbers are between 1000 and 2000, traversed in order. Safety factors are 1 to 5 (5 being "safe"). Names are simply descriptions of the treat.

    Your **Treat** class must have:

a. A **single constructor** that serves as both the **default** and **non-default** constructor. (default data: first house, received an "unsafe" treat of an apple)
b. Appropriate **accessors** for each data member
c. Appropriate **mutators** for each data member
d. An overloaded **insertion operator<<** that will display the information for this treat, allow this operator **direct** access to the data members.
e. <u>**3 data members**</u> that represent the **name, house number,** and **safety factor**
f. All minimum and maximum values for data members should be **constant**, **shared** data that is **inaccessible** to outside classes/functions

16. (5 pts) Implement the **constructor** for your Treat class, use other class methods when appropriate.

17. (5 pts) Implement the **mutator** for your **safety factor** data member, include code to verify the new value is within appropriate limits.

18. (5 pts) Implement the overloaded **insertion** operator for your Treat class

Aggregation

19. (15 pts) Declare a **BagOfTreats** class (again, do not implement, yet). Obviously, your BagOfTreats holds a **collection** of **Treats**. Think very carefully about your return types for this class (hint, hint).

Your BagOfTreats class must have the following:

g. A **default** constructor
h. A method to **add** a Treat to the bag.
i. A method to **remove** the i^th^ Treat from the bag, returning it to the calling function (i = 1 to size).
j. A method to **get** the i^th^ Treat from the bag, **without removing** it (i = 1 to size) or altering it.
k. An overloaded **operator+** to **combine** two bags together into a third (ex: BagOfTreats c = a + b;)
l. An overloaded **insertion operator<<** to print the entire **BagOfTreats** – this operator **should not** have direct access to the data members.
m. A **dynamic data member** to store a **collection** of Treats.

_________ pts

20. (5 pts) Implement the **add** method for your BagOfTreats class.

21. (5 pts) Implement the **remove** method for your BagOfTreats class.

22. (5 pts) Implement the **operator+** for your BagOfTreats class.

23. (5 pts) **Describe** how the **compiler** matches a function call to an **overloaded** function. Provide an **example** to **illustrate,** use of **prototypes** is acceptable, you can assume the implementation is elsewhere.

24. (5 pts) **Explain** why it is **better** to pass user-defined **objects** by **reference** than by value. **Provide** an **example** to **justify** your claim.

_________ pts

Extra Credit

25. (5 pts) **Overload** the **operator++** (postfix) for the **BagOfTreats** class. It should add a default Treat to the BagOfTreats. You may assume it is prototyped elsewhere.

26. (5 pts) **Develop** a **new** example using **static methods** and **static data** within a class. Your example **cannot** have been discussed in class or in the lecture notes or slides. Make sure that your example **demonstrates** your **understanding** of static methods and data. Do **not** just write a class **declaration**…

27. (3 pts) Describe one **unique** object in Ms. Wortman's office (this may not include the normal "office-type" items like: stapler, ruler, pen, pencil, computer, etc.).

28. (1 pt) Describe the **best Halloween costume** you have ever seen, worn, or thought of (particularly creative answers that make me laugh will get more points)?

29. (1 pt) What is your **absolutely favorite** type of **Treat**? **Trick**?

_________ pts