CMSC 202 Midterm March 16, 2006

Name: _____ **Email ID:**_____

(Circle your section)

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

105 – Tuesday 1:30 **104** – Thursday 12:30

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	
9 (EC)	**12**	
TOTAL	**100 (+12)**	

True/False (10 pts, 1 pts each)

Decide if the following are **true** or **false**; put the appropriate ***word*** in the blank.

_____ 1. When using command-line parameters, `argc` represents the index of the last item in `argv`.

_____ 2. When overloading functions, you can differentiate the function signature by return value. So, **`int foo()`** and **`double foo()`** are an example of valid function overloading.

_____ 3. The extraction operator cannot be overloaded as a member function of a user-defined class.

_____ 4. Only methods of a class can access private data members or methods within that class.

_____ 5. Class methods have access to the private data of *all* objects of that class, not just the current object.

_____ 6. The following code will print the character 'x' to standard out:

```
string message = "exit";
cout << message[2] << endl;
```

_____ 7. The following code is valid according to the ANSI standardized g++ compiler:

```
for (int i = 0; i < 10; ++i)
     cout << i << endl;
cout << "last i: " << i << endl;
```

_____ 8. The following code will print: `**ABC`

```
cout << setfill('*') << setw(5)
     << "ABC" << endl;
```

_____ 9. User-defined classes, in most situations, should be passed by reference or const reference to functions or class methods.

_____ 10. Static data members can only be modified by static methods of a class.

_____ pts

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) Declare an **input** stream and o**pen** a **file** named "activities.txt".

12. (2 pts) Declare **a vector** of **strings** to store the Spring Break **activity descriptions**. Declare a **vector** of **integers** to store the **number of people** at each activity.

13. (6 pts) Use a **loop** to **read** activity data from the **file**. **Store** the values in the above **vectors**. You can **assume** the file is **correctly formatted** as follows:

Ex:
```
35 Tour of Cancun City
23 Ski trip to Colorado
```

14. (10 pts) **Implement** a function that **accepts** the above **vectors** as **parameters**, **displays** trip and number of attendees, and **calculates** and **prints** the total attendees. Data should be printed in a **tabular format**, assume descriptions are **< 40 characters**.

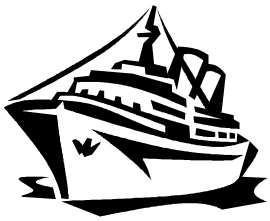
Ex:
```
Tour of Cancun City           35
Ski trip to Colorado          23
--------------------------------
TOTAL                         58
```


_____ pts

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) You are planning your **Spring Break trip** and need to organize all of the **events** that you plan on doing while on vacation. **Design a class** to represent a single **Event** (header only, implementation is the next page). Each event has a **start day** (numerical), a **start month** (numerical), and a **description**. **Valid months** are between **1 and 12**. **Valid days** are between **1 and 31** (do not worry about shorter months).



Your **Event** class must have:

a. A **single constructor** that serves as both the **default** and **non-default** constructor. (default date: January, 1, event description is "Default Event")
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 Event, this operator should **not** have **direct** access to the data members.
e. **3 data members** that represent the **month, day and description**
f. All minimum and maximum values for data members should be **constant**, **shared** data that is **inaccessible** to outside classes/functions

_____ pts

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

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

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

_____ pts

Aggregation

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

Your **Vacation** class must have the following:

a. A **default** constructor
b. A method to **add** an Event to the Vacation.
c. A method to **remove** the i^{th} Event from the Vacation, **returning it** to the calling function (i = 1 to size).
d. A method to **get (GetSize)** the **number** of Events in the Vacation.
e. A method to **get (GetEvent)** the **index** of an Event from the Vacation, that **matches** the date supplied in the parameter (the month followed by the day, example: today would be (3, 16)).
f. An overloaded **operator-** to **remove** the i^{th} Event from the Vacation, without changing the original object (ex: int j = 2; Vacation c = a - j;)
g. An overloaded **insertion operator<<** to print the entire Vacation – this operator **should** have **direct** access to the data members.
h. A **dynamic data member** to store a **collection** of Events.

_____ pts

20. (5 pts) Implement the **remove** method for your Vacation class.

21. (5 pts) Implement the **GetEvent method** for your Vacation class. Hint: loop through all of the events looking for a match....

22. (5 pts) Implement the **operator-** for your Vacation class.

_____ pts

23. (5 pts) **Describe** the idea behind a **Zombie** object. Provide an **example** that demonstrates the **necessity** of Zombie objects (Hint: describe an instance where there is no other way to communicate…).

24. (5 pts) Define **encapsulation**. Provide an **example** that demonstrates the **power** of encapsulation.

_____ pts

Extra Credit

25. (5 pts) **Implement** the **add** method for your Vacation. It should **insert** a new Event by maintaining a list of **sorted** Events (i.e. insert the new Event in **date-sorted** order).

26. (5 pts) **Describe** the potential problem when using both `getline( )` and the **extraction operator >>** to **read** strings and integers. Provide an **example** to **support** your argument. **Demonstrate** a method for **solving** this problem.

27. (2 pts) If you could pick **anywhere** on the **planet** to visit for Spring Break, **where** would it be and **why**?

_____ pts