University of Maryland
Baltimore County

CMSC 202 – Computer Science II

Fall 2005

Mid-Term Exam

Sections 0201 – 0206

Lecture Hours: Monday – Wednesday 5:30 PM – 6:45 PM

Exam Date: Wednesday 3/16/05

Exam Duration: 5:30 PM – 6:45 PM

Instructor: Sa'ad Raouf

Name:	
SSN:	
Score:	
Section: (Circle your Section Number below)	

Section	TA	Day	Time	Room
0201	Rick, Jim	Monday	7:30 - 8:20pm	ECS 104 A
0202	Jim	Wednesday	7:30 - 8:20pm	ECS 104 A
0203	Jim (, Deep)	Monday	2:30 - 3:20pm	ECS 104 A
0204	Jim	Wednesday	2:30 - 3:20pm	ECS 104 A
0205	Rick	Monday	11:00 - 11:50am	ECS 104 A
0206	Deep	Wednesday	11:00 - 11:50am	ECS 104 A

Notes:
1. This exam is a closed book, and a closed notes exam.
2. All answers are to be written on the enclosed exam sheets. Scratch sheets are not allowed. If necessary, you can use the back of the exam sheets.
3. You will need to present your Photo ID when handing in the exam. No exceptions.
4. Please hand in your exam with your section number circled. If your section number is not circled, your exam will not be graded.

1) (2 points each) Write True or False in the TRUE/FALSE column:

		TRUE/FALSE
a)	>> is called the stream insertion operator	**FALSE**
b)	vector <int> test[12]; creates a vector of 12 integers	**FALSE**
c)	When the following statement is executed students.push_back(aJake); a copy of aJake is pushed into the students vector.	**TRUE**
d)	Raouf says: "Use references when you need to, use pointers when you have to"	**TRUE**
e)	Raouf says: "It's OK to have a function with multiple return statements, instead of only one return statement"	**TRUE**
f)	Raouf suggests: Prefix your class name with a capital C. (this one is for you, Tony!)	**TRUE**
g)	Raouf declares: Keep the class logic together with the class interface.	**FALSE**
h)	A class can have more than 1 destructor	**FALSE**
i)	Static data members of a class are created when the first instance of the class is constructed, and the static data members are destroyed when the last instance of the class goes out of scope	**FALSE**
j)	operator<< can be overloaded as a class member function	**FALSE**

2. **(5 points)** Declare a vector of vector of vector of integers called cube.

Answer:

vector < vector < vector <int> > > cube;

3. **(5 points)** Explain and show what the following code will produce as output*.*

```
vector <int> integers(5,99);

unsigned int  size = integers.size( );
for (unsigned int i = 0; i < size ; i++)
{
      cout << setfill('*') <<setw(5) <<integers.at( i ) << endl;
      integers.erase(integers.begin());
}
```

Answer:

```
***99
***99
***99
```

And then segment fault

4. **(5 points)** Write one single C++ statement to modify the variable declared below such that the dashes are removed from the string.

string ssNumber = "555-78-1234";

Answer:

```
            ssNumber = ssNumber.substr(0,3) + ssNumber.substr(4,2) +
ssNumber.substr(7,4);
```

5. **(5 points)** Write the missing C++ statement from the code snippet below. Output an appropriate error message if any errors occurred. The code simply reads in one or more strings from an input file, and outputs the strings to an output file. If opening the input file fails, then exit with a status of -1. If opening the output file fails, then exit with a status of -2.
   Assume that all needed header files have been included.

   Answer:

```
//there are missing C++ statements here




ifstream input;
      ofstream output;

     input.open( "infile.txt" );
     if (input.fail( ) )
     {
        cout << "Error opening infile.txt" << endl;
      exit( -1 );
     }

     output.open("outfile.txt");
     if (output.fail( ) )
     {
        cout <<"Error opening outfile.txt" << endl;
            exit( -2 );
     }

      string aString;
      while (input >> aString)
      {
            output << aString;
      }
      input.close();
      output.close();
```

NOTE TO GRADER:

- **No partial credit here**
- **Any text in the error message is acceptable**
- **It is acceptable, if the code prompted for an input or output file name, storing the file name in a string type, and then using input.open(someString.c_str();**

6. **(5 points)** Why do we have to use a C-style string for opening an input file stream:

Answer:

History: The iostream libraries/code was developed before the string class was developed, and as such support for the strings class is not included in the iostream libraries/code

NOTE TO GRADER:
This is a vague question, so you can be flexible here. Use your judgement.

7. **(6 points)** Write the C++ function that would center a string in a field width of n characters wide. Both the string and the width are passed to the function. Name the function as **center**, name the input string parameter as **aString**, and name the width parameter as **width**. Write the appropriate function pre-conditions, and post conditions to match your logic. Assume that all appropriate header files have been included.

Answer:

```
void center (string aString; int width)
{

      int n = (width - aString.length()) / 2;
      cout << setw(n)<< ' ';
      cout <<aString;
      cout << setw(n)<< ' ';
}
```

NOTE TO GRADER:
Some students might have returned a string from this function, that is acceptable. Also, there are many ways that this can be solved, so be flexible. 1 point or the function pre-condition, 1 point for the function post-condition. 3 points for the code. The pre-condition, and post-condition must match the logic. Example would be how to handle the string being greater than the width.
MAKE THIS QUESTION COUNT FOR 5 POINTS!!!!

8. **(5 points)** Explain each of the following function prototypes in terms of what can and cannot be done to the aVector parameter within the Initialize function:

   a) void Initialize (vector <int>   aVector);
   b) void Initialize (const vector <int> aVector);
   c) void Initialize (const vector <int> & aVector);
   d) void Initialize (vector <int> & aVector);

   Answer:
   a) void Initialize (vector <int>   aVector);
   Pass by value, aVector can be changed in function Initialize, changes will not be reflected back in calling function
   b) void Initialize (const vector <int> aVector);
   Pass by const value, aVector can not be changed in function Initialize, No changes will be reflected back in calling function
   c) void Initialize (const vector <int> & aVector);
   Pass by const reference , aVector cannot be changed in function Initialize, No changes will be reflected back in calling function
   d) void Initialize (vector <int> & aVector);
   Pass by reference, aVector can be changed in function Initialize, changes will be reflected back in calling function

NOTE TO GRADER:
Make this question count for 4 points only, 1 point for each correct answer. Use your judgement, but if pass by value, and pass by reference is mentioned, than they are probably on the right track.

9. **(5 points)** Given the following mutator method:

   {some return type here} DayOfYear::Set(int newMonth)

   to set the class member variable m_month to a value that should only be between 1 and 12. List 6 things that you can do to mitigate the condition where the newMonth value is not between 1 and 12.

   Answer:
   a. reprompt for a newMonth
   b. output an error message
   c. set the m_month = 0;
   d. return false;
   e. zombie object
   f. throw an exception

Note to grader:
Make this question count for 6 points. Partial credit is OK. 1 point for each item listed.

10. **(5 points)** Explain the concept of and implementation of zombie objects.
    Answer:
    Create a private Boolean data member (m_isValid) that is set to true if the parameter data in a mutator or a constructor is correct, and set to false otherwise.
    Create a public accessor method called is_valid () to return the value of the Boolean data member
    Create a private member function to perform the validation. This function will be called from the constructor, or from within all mutator methods.

NOTE TO GRADER:
Use your judgment, but we need to see a private data member, and an interrogation function that returns the value of the private data member.

11. **(5 points)** Given that Toy is a user defined data type, explain each of the following statements:

    a) Toy myToy();
    b) Toy myToy;
    c) Toy myToy (GIJoe);
    d) Toy myToy [4];

    Answer:

    a) Toy myToy();
    2 points. A function prototype for a function that returns a Toy object.

    b) Toy myToy;
    1 point. A declaration (default constructor call) for a variable called myToy that is of type Toy.

    c) Toy myToy (GIJoe);

    A declaration (non- default constructor call) for a variable called myToy that is of type Toy. GIJoe can be any data type, or a Toy type

    d) Toy myToy [4];
    1 point. A declaration for a myToy variable that is an array of 4 Toy objects

12. **(5 points)** Write the function prototype to overload the operator < for class Money, as:
    i. a member method.
    ii. a non-member method

Answer:

i. a member method.

```
bool operator< (const Money& RHS);
```

ii. a non-member method

```
bool operator< (const Money& LHS, const Money& RHS);
```

13. **(5 points)** Now write the function for the previous question, assuming that the money class has 2 member variables, called m_dollars, and m_cents, and that the needed accessors for m_dollars, and m_cents have **not** been defined

Answer:

```
bool Money::operator< (const Money& RHS)
{
      if (m_dollars < RHS.m_dollars)
            return true;
      else
            if (m_dollars  = RHS.m_dollars && m_cents < RHS.m_cents)
                  return true;
            else
                  return false;
}
```

Note to Graders:
2 points for using a member function.
3 points for the code

(10 points) Rewrite the following class definition, using const and reference to const where applicable. Assume that the convention of a member function name starting with **set** is a mutator, and the member function name starting with **get** is an accessor.

```
const string Car::m_className = “Car”;
    class Car
    {
       public:
            Car(int passengers = 5,  const string& color = "Red");
            string& GetColor (  );
            int GetNrPassengers(  );
            void SetColor (string& color);
            void SetNrPassengers (int passengers);
    private:
            int    m_passengers;
            string m_color;
            bool   m_isMoving;
            static string m_className;
    };
```

Answer:

```
const string Car::m_className -= “Car”;
class Car
{
     public:
    Car(int passengers = 5, const string& color = "Red");
  const string& GetColor (  ) const;
  int GetNrPassengers(  ) const;
  void SetColor (const string& color);
  void SetNrPassengers (int passengers);
private:
  int    m_cylinders;
  int    m_passengers;
  string m_color;
  bool   m_isMoving;
  static const string m_className;
};
```

1 point for each const.

(5 points) Given the class defined above, write the constructor using a member initialization list

Answer:

```
      Car::Car(int passengers,  const string& color):

   m_passengers (passengers),
   m_color (color)
      {
      // no code
      }
```

14. **(5 points)** Given the following constructor, write the destructor

```
// Car constructor

Car::Car( const string& color, int miles )
{
   m_color = new string;
   *m_color = color;

   m_mileage = new int( miles );
}
```

Answer:

```
// Car's destructor

Car::~Car(  )
{
   delete m_color;
   delete m_mileage;
}
```

Note to grader: No partial credit.
Not setting the pointers to 0 is OK, do not deduct points for not setting the pointer back to 0.