University of Maryland
Baltimore County

CMSC 202 – Computer Science II

Fall 2004

Mid-Term Exam

Sections 0201 – 0206

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

Exam Date: Wednesday 10/20/2004

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	Patrick & Carrie	Monday	7:30 - 8:20pm	ECS 104 A
0202	Patrick & Carrie	Wednesday	7:30 - 8:20pm	ECS 104 A
0203	Patrick & Ryan	Monday	2:00 - 2:50pm	ECS 104 A
0204	Patrick & Ryan	Wed	2:00 - 2:50pm	ECS 104 A
0205	Patrick & Carrie	Monday	11:00 - 11:50am	ECS 104 A
0206	Patrick & Carrie	Wed	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)	argv[1] is always the name of the executable program	
b)	Class mutator methods can only be declared as const methods	
c)	If a mutator method receives invalid data, then the mutator should terminate the program.	
d)	The following vector declaration causes the constructor for the Student class to be invoked: vector <Student> myStudents;	
e)	In the method for the post increment operator for the Money class: **Money operator++(int dummy);** the value of the dummy parameter is zero.	
f)	In C++, the stream extraction operator is defined as **operator>>**.	
g)	endl and \n are the same, there is no difference between using endl and \n in a cout statement	
h)	A static data member of a class is considered a “global” data member for all instances of the class	
i)	Class methods cannot be declared as static methods.	
j)	Mr. Raouf mentioned in the lecture that operator overloading is an example of **Syntactic Sugar**	

2. **(6 points)** List 3 issues where vectors are superior to arrays. Give an example for each issue. Use the array declaration int myArray[10]; for your examples.

Answer:

3. **(4 points)** Show the output from the following code snippet:

```
vector < int > integers (5);
for (unsigned int i = 0; i < integers.size( ); i++)
{
            cout << integers.at( i ) << endl;
             integers.push_back(i);
}
```

- What will the value of i be after the for loop has completed?
- What will the size of the vector be after the for loop has completed?

Answer:

4. **(4 points)** Rewrite the following code to use call by reference parameters instead of pointers.

```
// function prototype
void add1 (int * pMyInteger);

//function definition
void add1 (int * pMyInteger)
{
      (* pMyInteger)++;

}
```

Answer:

5. **(6 points)** Write the C++ statements for the following:

- Create a vector of Bicycle objects called myBicycles
- Create an instance of a Bicycle class called someBicycle, using the default constructor.
- Insert the someBicycle object at the end of the myBicycles vector.

6. **(5 points)** Given that Circle is a class that contains a Point class, and you noticed the following constructor for the Circle class during your visit to the C++ Country Club:

   Circle (Point center, float radius);

   Explain if you should speak out, or remain silent, and why?

   ANSWER:

7. **(5 points)** Given the following function header for a class accessor called GetDayOfYear, explain the significance of each const listed below:

```
const DayOfYear  &  Vacation::GetDayOfYear ( ) const
```

answer:

8. **(6 points)** What will the following declarations do:

a. vector <Tractor * > machineShed;

b. vector < vector < string> > Words;

c. vector <char> WordsInAlphabet(26);

Answer:

a.	
b.	
c	

9. **(4 points)** Given the following class definition, write the statement to initialize static data member m_turn to 0;

```
class Server
{
   public:

   private:

      static int  m_turn;
};
```

Answer:

10. **(5 points)** Rewrite the following constructor to use a member initialization list

```
someClass::someClass( string  someName, float someMoney )
{
        m_Name = someName;
        m_Money= someMoney;
}
```

Answer:

11. **(5 points)** Since there is no way to prevent the user of a class from passing invalid parameter values to a constructor, define a mechanism that can indicate to the user of the class if invalid parameters were passed to the constructor.

ANSWER:

12. **(5 points)** Given that Toy is a user defined data type, explain each of the following statements:

- Toy myToy();
- Toy myToy;

ANSWER:

13. During one of the lectures where object oriented concepts were first introduced to you, we discussed a conceptual design for a Command Line class. The following is a class definition for CCommandLine class:

```
class CCommandLine
{
public:
        CCommandLine();
        CCommandLine(int argc, char *  argv[]);
        CCommandLine(int argc, const vector <string> & argv);
        string getParameter(unsigned int  index);
        bool findParameter(string parameter);
        unsigned int getParameterCount();
        vector <string>  getAllParameters();
        const CCommandLine  operator+(const string & LHS);
private:
        int m_argc;
        vector <string> m_argv;

};
```

The class CCommandLine is used to model argc, and argv as defined in main(). The class designer has decided to use 2 private data members:

- int m_argc: used to represent the number of arguments, same as the argc in main
- vector < string > m_argv: used to store the command line parameters, where each parameter is a string, same as the char * argv[] in main.

Given the above class definition, write the code for the following methods:

- **(5 points)** CCommandLine(): default constructor, use a member initialization list where applicable

- **(5 points)** CCommandLine(int argc, char * argv[]): constructor, use a member initialization list where applicable

- **(5 points)** CCommandLine(int argc, const vector <string> & argv): constructor, use a member initialization list where applicable

- **(5 points)** bool findParameter(string parameter): a method that returns true if the parameter is found in the parameter list, else returns false

- **(5 points)** const CCommandLine operator+(const string & LHS): overloaded + operator to add a string to the m_argv, and to increment m_argc;