CS 172 Review for Exam 1 (Spring 2017)

Exam 1 is a hands-on practical test. You are allowed to look at any previous programming solution for the exam, your book, slides, and notes. Also, if you are using the **string** class, you are allowed to look at http://www.cplusplus.com/reference/string/string/. The exam will cover **chapters 9 and 10**, but it will assume you are competent with all concepts covered in CS-171. **Please read the textbook** and **review the concepts** described in this document. You will have time in class to complete the exam and if you still need time to complete it, you may complete the exam by midnight.

C++ Review (all concepts covered in CS-171)

- Know how to write **for** and **while** loops, and **if** statements.
- Know how to define functions with parameters.
- Know how to define and use arrays
- Know how to specify mathematical and boolean expressions with the standard C++ operators, use the pseudo-random number generator, etc.

Designing Software with Objects

- Controlling complexity on large projects is a challenge. Know how Object Oriented design help us control complexity?
- How do you identify objects, properties and behaviors from a software requirements specification?
- What are the class access control keywords? E.g. **private** and **public**. What are they used for?
    - Know that there is a third keyword, **protected**, but you will not be tested on this.
- How do you define a member variable (i.e. property) in C++? How do you define a member function (i.e. behavior) in C++?
- A class is blueprint (or template) for making ___________________.
- What is the purpose of the **'.'** operator?
- What is the purpose of a class **constructor** and how do you **define one**? Know how to define **overloaded constructors**.
- Know what is a header **inclusion guard** and how to define them
- Know how to put a class declaration in a **class definition header file** (*.h file), and put method implementations in a **class implementation file** (*.cpp file)
- Know how to read and interpret **UML** class diagrams.

The string Class - An Example of a Good O.O. Design

- Know how to use the methods available in the string class:
    - You will be allowed to refer to http://www.cplusplus.com/reference/string/string/ if you need too.

Passing Objects to Functions, Arrays of Objects, Modeling Object Composition Relationships

- What is the difference between function parameters that pass an object "**by value**" and those that pass an object "**by reference**"?
- What is the purpose of **const**? Why is it useful and where can it be used?
- What are **static** member variables/methods?
- Know how to compose 2 classes together. For example, when the software requirements say that a **Student** *has-a* **Faculty** advisor, this will result in a design where a **Student** class contains a member variable of a **Faculty** class type. Know how to read and interpret **composition (HAS-A) relationships** in **UML** diagrams.
- How do you define an **array of objects**? How do you **access** individual object elements in the array? How do you **initialize** an array of objects?

Practice Exercise for Exam 1

- Create a github repo called CS172-Exam1-Review. Push all of the code to that repo. You will turn in your exam by submitting your code to github.

- Define a class called **Dice** and put this in a class definition file called **Dice.h**.
    - This class will simulate a die object with a user specified number of sides.
- The **Dice** class should have **a constructor** with one parameter that defines the number of sides of the die as an argument.
- The **Dice** class should also have the following **public functions** (behaviors) implemented in a file called **Dice.cpp**:

```
Dice(int Sides);   // The constructor sets the number of sides of the die.
                   // The constructor also calls srand() to initialize
                   // the random number generator.
int Roll();        // Returns the results of a roll
                   // i.e. a random value between 1 and the number of sides of the die
int GetRolls();// Returns the number of times this dice was rolled since it was created.
int GetSides();    // Returns the number of sides for this dice object.
```

- Think about what additional **private properties** the **Dice** class needs to have in order to correctly implement the above behaviors. *Hint: remember a dice needs to remember the number of sides it has and the number of rolls.*
- Once your class is designed and written, test it with the following program:

```
#include <iostream>
#include "Dice.h" // Include your Dice class here
using namespace std;

int main()
{
      Dice d(6);                    // Declare a dice with 6 sides
      cout << d.Roll() << endl;     // Outputs a value from 1 to 6
      if (d.GetSides() != 6) {
            cout << "Error in GetSides().  It should return 6\n";
            return 0;
      }
      // Roll the dice 100 times and make sure it works every time.
      bool passed = true;
      for (int i = 0; i < 100; i++) {
            int x = d.Roll();
            if ( x < 1 || x > 6 ) { // Roll returned an incorrect side
                  cout << "Error in Roll() method! Roll returned " << x << endl;
                  passed = false;
                  break;
            }
      }
      if ( passed )
            cout << "Passed roll test" << endl; // Should output this message!
      cout << d.GetRolls() << endl;   // Should output 101
}
```

Hint for this practice only:

1. To use the pseudo random number generator, you need to include the files
```
#include <ctime>
#include <cstdlib>
```
2. To initialize the pseudo-random number generator, you need a statement like
```
srand(time(NULL));
```
3. To get a pseudo-random number, you can call **`rand()`** which will return an integer number.