2143 - OOP
Fall 2021
Exam 1
September 27, 2021

Name: _____

READ THESE INSTRUCTIONS
• Use pencil only • Initial top right corner of all pages (except the first one). • Do not remove the staple from your exam. • Do not crumple or fold your exam. • Handwriting that is illegible (messy, small, not straight) will lose points. • Indentation matters. Keep code aligned correctly. • Answer all questions in the provided space directly on the test. • If your answer will not fit in the space (IT SHOULD) use the blank sheets at the end of the exam. Write *"On Back"* at end of question and label that question clearly on the back sheets. • Help me ... help you!
Failure to comply will result in loss of letter grade

Grade Table (don't write on it)

Question	Points	Score
1	20	
2	10	
3	20	
4	15	
5	15	
6	20	
Total:	100	

1. (20 points) There are 3 major concepts when we think about OOP. What are they? Use layman's terms to define each word, give examples if possible. *List your answers in alphabetical order*.

Answer:

For this exam, just a basic idea of each term will suffice, expecting more detailed understanding as the semester goes on.

Concept 1: **Encapsulation**

Packaging together the data along with the methods that work on the data.

Typically confused with **abstraction**. One way to separate the two definitions is (in general) use *Encapsulation* when discussing implementation (or code) and use *Abstraction* when discussing design.

Concept 2: **Inheritance**

The concept of inheritance put a few different ways.

- Defining one class in terms of a another class.
- One class deriving properties and characteristics from another class.
- Using previously designed class or classes to build a current class. Done by incorporating the *parent* or *base* classes methods and data into a *sub* or *child* class along with additional functionality tailoring the current class to a more specific need.

Concept 3: **Polymorphism**

Polymorphism means to have "many forms" and has inherently many forms in which it shows itself. I didn't expect near this level of detail in your answer, but will shortly.

- Compile Time Polymorphism
  - Function Overloading: We can define a method of the same name with different parameter lists.
  - Operator Overloading: Defining what should happen when a class is used with a specific operator.
- Run Time Polymorphism: To take on a form during runtime depending on .... will talk about it later.

2. (10 points) What is the main difference between a class and an object?

Word Choices				
A. Code	**B.** Definition	**C.** Implementation	**D.** Memory	**E.** State

Answer:
Use the corresponding letters for the words above to fill in the blanks below.

A **Class** has a **Definition** , and **Implementation**

whereas
an **Object** resides in **Memory** and has **State** .

---

3. (20 points) Write a class **definition** called **RgbColor** to represent an RGB color. Even if we haven't discussed what an RGB color is, it is pretty intuitive. It contains 3 values, 1 for each of the **R**ed, **G**reen, and **B**lue, color channels. A valid value for each channel is between 0-255 inclusive. Examples:

- Red : [255,0,0] Red max value. Green zero. Blue zero.
- Purple : [255,0,255] Red max value. Green zero, Blue max value.
- Yellow : [255,255,0] Red max value. Green max value. Blue zero.

Methods:

- This class should have a setter and getter for each data member. If this term confuses you, it means there should be one method per data member to SET it (change it) and one method per data member to GET it (return what it is). Naming these methods should be similar to: *SetRed* and *GetRed*.

- Add a method prototype called "GrayScale" that would return a grayscaled version of the color currently in the class. Gray-scale just means average the 3 color channels, and replace each channel with the averaged value. Example red [255,0,0] becomes [85,85,85].

Usage: Look at the usage to determine *other things* that may go in your class *definition*:

```
1   RgbColor Color1;                // each color = 0
2   RgbColor Color2(0.30,1.0,0.50); // each color a percentage of 255
3   RgbColor Color3(0,200,0);       // each color as passed in
```

Answer On Next Page...

Answer:

```
class RgbColor{
    private:
    int r;
    int g;
    int b;

    public:
    RgbColor();
    RgbColor(int , int , int );
    RgbColor(double  , double , double );
    void setRed(int );
    void setBlue(int );
    void setGreen(int );
    int getRed();
    int getBlue();
    int getGreen();

    RgbColor grayscale();
};
```

4. (15 points) Overload **ostream** for the *RgbColor* class so it prints the color values with the following format: **[r,g,b]** (replacing r,g,b with the objects numeric values). Assume you are defining this within the class definition (inline).

Answer:

```
friend ostream& operator<<(ostream& os, const RgbColor rgb){
    os<<"["<<rgb.r<<","<<rgb.g<<","<<rgb.b<<"]";
    return os;
}
```

5. (15 points) Overload the subtraction **-** operator for our *RgbColor* class so it calculates the difference of two colors. RGB values cannot be negative, so assume we have included **math.h** and use the **abs** function. Look at example below:

```
RgbColor Color1(200,125,20);
RgbColor Color2(150,100,40);
RgbColor Color3 = Color1 - Color2; // Color3 = [50,25,20]
```

Again, assume you are defining this within the class definition (inline).

Answer:

```
RgbColor operator-(const RgbColor &rhs){

    return RgbColor(abs(this.r - rhs.r) , abs(this.r - rhs.r) , abs(this.b - rhs.b));
}
```

6. (20 Points) Copy constructor and Assignment Operator questions.

(a) (5 points) Do we need a copy constructor for every class? Explain.

Answer:

No we do not. The main reason to create a copy constructor is when dynamic memory is involved. A shallow memberwise copy (which gets created by the system for you) is not sufficient when dynamic memory is involved, and the programmer needs to ensure new memory is allocated for the new object. If we don't allocate new memory, both object will end up pointing to the same memory!

(b) (5 points) Does the RgbColor class need a copy constructor? Explain.

Answer:

No! There is no dynamic memory involved, so the default copy constructor created by the system is good enough.

(c) (5 points) Do we need to overload the assignment operator (=) for every class? Explain.

Answer:

No! Its the same situation as the copy constructor. If we don't have dynamic memory (pointers) then the system created assignment operator will suffice.

(d) (5 points) What do we need to check for when overloading the assignment operator? Explain.

Answer:

Check for self assignment. Overwriting "this" with itself has very bad ramifications. We typically "destroy" the copy we are overwriting first, and then overwrite it with the incoming copy. This works well for "other" objects passed in to the assignment method. However, if we pass a copy of our-self to the assignment method, then follow the "destroy the local copy" mentality, we end up destroying everything since both copies are the same!