2143 - OOP
Spring 2021
Exam 1
April 26, 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	10	
2	15	
3	18	
4	20	
5	20	
6	15	
7	15	
8	20	
9	20	
10	10	
Total:	163	

Observation:

My first observation when grading your exams was that many of you did not use proper terminology when "defining" or explaining concepts. It is OK in some contexts, but not too many. Words are not interchangeable especially when I am trying to be as objective as I can while trying to grade a subjective exam. So, a class is not a **blueprint**, or a **skeleton**, or a **recipe**, it is a "**definition**" of an **Abstract Data Type**.

1. (10 points) What is the main difference between a class and an object?.

Answer:

A **Class** is the **definition** of an Abstract Data Type (**ADT**) which includes methods and data members.

An **Object** is an **instance** of some ADT (it has been declared). This means it now *resides in memory and has state.*

---

2. (15 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.*

Remember: Layman's terms....

Concept 1: Encapsulation

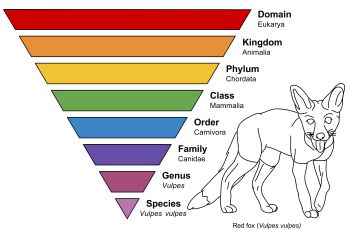
Definition: Packaging data and methods together in a single construct (aka ADT).

Example: The hierarchy of biological classification's is something that would fit with the concept of inheritance. Top of the hierarchy are broad descriptions and classifications getting more detailed as you move down the ranks.

Concept 2: Inheritance

Definition: Is basing an object or class (child) upon another object or class (parent) The *child* gains all the functionality of the *parent*, plus a little (or a lot) more targeting a more specified use for the child class.

Example:



Concept 3: Polymorphism

Definition: More than one form.

Examples: (just need to list one of the following)

1. Function Overloading
2. Operator Overloading
3. Function Overriding
4. Virtual Functions

3. (18 points) Given the list of definitions, match it with the correct word. ~~The same word can be used more than once~~. All words don't have to be use. *Use the corresponding LETTER next to the word choice to fill in your answer.*

#	Letter	Definition
1.	**E**	It is a special type of subroutine called to initialize an object.
2.	**F**	Cleans up allocated memory for a class.
3.	**I**	Defining class A by basing it on class B.
4.	**A**	Hiding the details of the implementation from the user.
5.	**G**	Packaging data and methods together.
6.	**H**	Used to allow access to private members in a class.
7.	**P**	Overloading methods.
8.	**L**	A function, except its in a class.
9.	**C**	A variable that is shared by all instances of a class.
10.	**D**	Using instances of 1 or more classes as part of the definition of another class (like a data member).
11.	**O**	Same function name, different parameters.
12.	**Q**	Variables in this section cannot be read by sub classes.

Word Choices							
A	Abstraction	F	Destructor	K	Member-Variable	P	Polymorphism
B	Class	G	Encapsulation	L	Method	Q	Private
C	Class-Variable	H	Friends	M	Multiple-Inheritance	R	Protected
D	Composition	I	Inheritance	N	Object	S	Public
E	Constructor	J	Instance-Variable	O	Overloading	T	Virtual

4. (20 points) Label each with an **A** for abstraction or an **E** for encapsulation.

#	A or E	Description
1.	**A**	Hides certain methods from users of the class by protecting them or making them private.
2.	**E**	Hides whether an array or linked list is used.
3.	**E**	Solves problem at implementation level.
4.	**E**	Wraps code and data together.
5.	**A**	Is focused mainly on what should be done when designing a class.
6.	**E**	Is focused on how it should be done when coding a class.
7.	**A**	Helps developers to design projects more easily.
8.	**A**	Lets a developer use a class without worrying about how it's implemented.
9.	**A**	Solves problem at design level.
10.	**E**	Hides the irrelevant details found in the code.

---

5. (20 points) Write a class **definition** called **RgbColor** to represent an rgb color. Recall from class that an rgb color contains 3 values, 1 for each of the red, blue, and green color channels. The values can only go from 0-255 for each channel.

   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). One example of each would be: *SetRed* and *GetRed* (or similar).
   - Add a method called "GrayScale" to turn whatever rgb color that is currently in the class to its gray equivalent and returns an RgbColor type with the gray color in it. Remember, gray-scale just means average the 3 color channels.

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

```
RgbColor Color1;            // sets each channel to zero
RgbColor Color2(34);        // sets each color channel to same value
RgbColor Color3(0,200,0);   // sets each channel to specified color
```

   Remember ... only the class definition.

   Answer On Next Page...

Answer:

```
class RgbColor{
        int r;                  //int 0-255 red
        int g;                  //int 0-255 green
        int b;                  //int 0-255 blue
        public:
        RgbColor();             // default constructor
        RgbColor(int);          // overloaded 1
        RgbColor(int,int,int);  // overloaded 2

        void SetR(int);         // setter red
        void SetG(int);         // setter green
        void SetB(int);         // setter blue

        int GetR();             // getter red
        int GetG();             // getter green
        int GetB();             // getter blue

        RgbColor GrayScale();       // averages colors

        // print to stdout
        // not part of actual answer
        friend ostream& operator<<(ostream&,const RgbColor&);

        // add (mix) two colors
        // not part of actual answer
        RgbColor operator+(const RgbColor& );
};
```

6. (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, so no scope resolution operator necessary.

Answer:

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

7. (15 points) Overload the addition + operator for our *RgbColor* class so it averages the two colors. Look at example below:

```
RGB Color1(100,50,20);
RGB Color2(200,100,40);
RGB Color3 = Color1 + Color2; // Color3 = [150,75,30]
```

No need to overload the assignment operator. Assume you are defining this within the class definition.

Answer:

```
RgbColor RgbColor::operator+(const RgbColor& rhs){
    RgbColor temp;
    temp.r = (this->r + rhs.r) / 2;
    temp.g = (this->g + rhs.g) / 2;
    temp.b = (this->b + rhs.b) / 2;

    return temp;
}
```

8. (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! You really only need a copy constructor when there is dynamic memory involved and we need to do a **deep copy**. If there are only compile time variables local to the object, then there isn't a real need. If you do need to write a copy constructor, then you should adhere to the rule of three. We did NOT discuss this in class, but going over tests is a great learning moment.

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

Answer:

Well, based on the previous answer I would say **no**. There is no dynamic memory and three simple data members. Compiler will create copy constructor easy enough that will work.

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

Answer:

You may have guessed that all these questions are somewhat related. But to answer this question: NO. When you assign one variable of a certain data type to another of the same type, the compiler will (as in the copy constructor) make that happen for you. Problems will arise if we have dynamically allocated memory and need to do a "deep" copy. In this case, the same issues will arise. Let me refer again to the rule of three. I've never mentioned it, but we did talked separately about all these issues (hence the multi-part question :)).

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

Answer:

Simply put: **Self Assignment**. Remember the disaster that can happen if you don't? It's possible to destroy the object that the call is made on.

9. (20 points) Write a copy constructor for the following class definition.

```
class NumContainer{
        int *array;
        int size;
        public:
        NumContainer(){
                size = 100;
                array = new int[size];
        }
        NumContainer(int s){
                size = s;
                array = new int[size];
        }
        // More stuff we don't need to see ...
};
```

You can again assume that you are doing this within the class definition.

Answer:

```
/** Copy Constructor
*
*  Params:
*     const NumContainer& - the other object were copying from
*
*  Returns:
*     void
*/
NumContainer(const NumContainer& other){
        size = other.size;      // get size of other array


        array = new int[size];
        for(int i=0;i<size;i++){
                array[i] = other.array[i];
        }
}

//
```

10. (10 points) If there was one thing you studied, and I did not put it on the test, then write your own question, and answer it. Something too simple will not earn you points. It must be worthy....