Assignment #6 – Using Structs vs. Classes
Due: Friday, 06/07/13, 11:59pm

(75 points) You will alter you Assignment #5 and write a small program from the book.

(40 pts) First, begin by adding a struct to your Assignment #5. You will not be graded on the correctness of the program against Assignment #5 rubric. Instead, you will be graded on your program defining and using a struct, which compiles and runs. You will probably want to define a struct to hold all the information contained with the largest product found. For example:

```c
struct largest_product_info {
    int value;
    int row;
    int col;
    string shape;
};
```

(35 pts) Next, do problem 6.5 from the book. Define a class for a type called Fraction. This class is used to represent a ratio of two integers. Classes are very similar to structs, but they can contain member functions too:) For example, below is a class with the numerator and denominator:

```c
class Fraction {
    public:
        Fraction() { numerator = 1; denominator = 1; }
        int numerator;
        int denominator;
};
```

You can create a Fraction type just as you did with a struct, and then you can access the members just as you did with a struct.

```c
int main() {
    Fraction frac1;

    frac1.numerator = 20;
    frac1.denominator = 60;
    ...
}
```

Now, make the member variables private, and create mutator functions that allow the user to set the numerator and the denominator. Also, include a member function that returns the value of the numerator divided by the denominator as a double.
Include another member function that **outputs the value of the fraction reduced to the lowest terms.** For example, instead of outputting 20/60 the function should output 1/3. This requires finding the greatest common divisor for the numerator and denominator, and then dividing both by that number. Below is an outline of what your Fraction class should look like.

```cpp
class Fraction {
    public:
        Fraction() { numerator = 1; denominator = 1; }
        void set_numerator(int num);
        void set_denominator(int denom);
        void print_lowest_terms();
        double get_fraction_value();
    private:
        int numerator;
        int denominator;
};
```

Embed your class in a test program by making a Fraction object and calling each of the functions in the class. For example, you might write a test program that asks the user for an integer numerator and integer denominator, print the real number from evaluating the fraction, and then print the fraction reduced to its lowest terms.

**Enter a numerator:** 20  
**Enter a denominator:** 60  
**The result of your fraction is .333333.**  
**Your fraction in lowest terms is 1 / 3.**

**10 pts** In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Read the class style guideline for more information:  
[http://classes.engr.oregonstate.edu/eecs/spring2013/cs161-001/161_style_guideline.pdf](http://classes.engr.oregonstate.edu/eecs/spring2013/cs161-001/161_style_guideline.pdf)

---

You are graded on having a header, proper comments, and readable code with indentation and vertical spacing that is CONSISTENT throughout your program. DO NOT align your entire program on the left side. This will cause you to automatically lose the full 10 points. In addition, do not forget your program header!!!

**15 pts** You are required to turn in a written document **(as a pdf)** addressing Polya’s steps to solving a problem with step 3 being the C code you write to carry out/implement your plan. With this said, your written document must include these three sections:
Understanding the Problem
In your own words, explain what YOU think the problem is asking you to do. In this section, document your uncertainties about the problem and anything else that you feel was unclear or vague. This is to ensure that YOUR understanding matches MY understanding of the problem.

Devising a Plan/Design
At a minimum, provide an algorithm/pseudo code you designed to help solve the problem. In addition, include pictures/flow charts you used to help you devise your plan, as well as any other design decisions you made such as how to manage your time, how to decompose the problem, where to start first, etc. You can scan any handwritten work and attach it to the document as needed.

Looking Back/Self-Reflection
Report any checking/self-reflection you did while solving the problem. For instance, how did you make sense of the output from the implementation? This includes things such as using a calculator to make sure the output is correct, testing to make sure your code executes correctly and behaves the way you expect under specific circumstances, using external sources of information such as the internet to make sense of the results, etc. Also, include a statement about what you learned from the assignment.

Electronically submit your pdf document and two C++ program files by the assignment due date, using TEACH.