WEBD 236 Homework 6

Answer the following questions based on your reading of the text book, any supplemental material, and the instructor's presentation this week. If you use an external source (i.e. a web-page, the required textbook, or an additional book) to help you answer the questions, be sure to cite that source. You should probably always be citing a source.

Short answer

1. **[5 points]** Describe, in your own words, the 5 basic tenants of object-orientation.

Programming

2. **[3 points]** Write a function `reduce($arr, $func)` that takes an array and a function as a parameter. The reduce function should apply the parameter function to each element of the array in succession to produce a single result. Note: the current result should always be the first parameter to the function and the next element of the array should always be the second parameter. You may not use the PHP function `array_reduce` in your solution. For example, the result of the following should be 10.

```php
function myMax($current, $new) {
    return $current < $new ? $new : $current;
}

$arr = array(10, 5, 3, 5, 1, 2, 5, 7, 4);
print("Max: " . reduce($arr, 'myMax') . "<br />");
```

3. **[5 points]** Write a function `modeMaker()` that forms a closure such that the function it returns can be used with your `reduce()` function above to find the mode of an array. The mode of an array is the value that appears the most frequently (so in the `$arr` above, the mode is 5). To do this, you will need an array called `$seen` that keeps the count of times an element has been examined At the end of the reduce function, the `$seen` array should look like the following for the `$arr` above:

```
Array
(
    [10] => 1
    [5] => 3
    [3] => 1
    [1] => 1
    [2] => 1
    [7] => 1
)
```

Note that the closure you generate should always be returning the current mode for what it has seen so far (so it will either be the current mode, or the new element passed in). The following is a start for `modeMaker`:

```
function modeMaker() {
    $seen = array();
    return function($current, $new) use (&$seen) {
        // your code that uses $seen goes here
    };
}

$mode = modeMaker();
$arr = array(10, 5, 3, 5, 1, 2, 5, 7, 4);
print("Mode: " . reduce($arr, $mode) . "<br />");
```

4. **[12 points]** Write a class that models the concept of a Car that has a fuel economy (in miles per gallon), an odometer, and a gas tank. Your car should be able to drive a certain number of miles (specified as a parameter) until it reaches its distance or runs out of gas, whichever comes first. Further, you should write functionality to add gas to the car, and read the fuel gauge and the odometer. You should write the following methods:

   a. `__construct($initialGas, $mpg)`: which takes the initial gas amount and miles per gallon as parameters.
   b. `drive($miles)`: which will drive the specified miles (deducting gas from the tank and incrementing miles on the odometer)
   c. `addGas($gallons)`: which will add more gas to the tank (the tank is of unlimited capacity).
   d. `readFuelGauge()`: which will return the number of gallons of gas remaining in the tank.
   e. `readOdometer()`: which will return the number of miles logged on the odometer.

   You can use the following method inside the Car class to help you debug:

```
    public function __toString() {
        return 'Car (gas: ' . $this->readFuelGauge() .
            ', miles: ' . $this->readOdometer() . ')';
    }
```

   Whenever you print a car object, PHP will automatically call the `__toString()` method to determine what should be output.

If you execute the following simple program, you should get the output that follows:

```
$car = new Car(20, 25);
$car -> drive(25);
print($car . '<br />');
$car -> drive(1000);
print($car . '<br />');
$car -> addGas(5);
$car -> drive(10);
print($car . '<br />');
```

```
Car (gas: 19, miles: 25)
Car (gas: 0, miles: 500)
Car (gas: 4.6, miles: 510)
```

Submission instructions

Create a single Word document with the solutions to the short answer questions. Create a single PHP file with your solutions to the programming problems along with suitable test cases that demonstrate that they work. Follow proper coding conventions (indentation, commenting, etc.) that you would have learned in previous courses.

Submit both the PHP file and the Word document to the dropbox for the course.