Principles of Programming Languages Project 1
Dr. Li Aleha Crumpton

<u>Short Answer Questions</u>

#1. puts “Hello World”

#2. Ruby supports hash, array, set, and queue data structures. An array is basically an integer-indexed, ordered container of objects. A hash has the same concept as an array, but instead of being indexed with integers it is indexed by keys that you specify. A queue is a container for objects that are stored in the same order they were added into the queue, and are removed from the queue in that same order. That is, the first object added to a queue is the first object to be removed. A set combines the functionality of an array and storage methods of a hash and is an unordered set of objects that does not allow duplicates.

#3.

```
num_arr = []
for num in 1..10
    num_arr[num-1] = num
end
puts "#{num_arr}"
```

#4.

```
test = {'first_name' => 'Aleha', 'last_name' => 'Crumpton', 'tiger_email_id' => 'aic0002','banner_id' => '903670419', 'fav_movies' => 'O Brother, Where Art Thou?'}
puts test
```

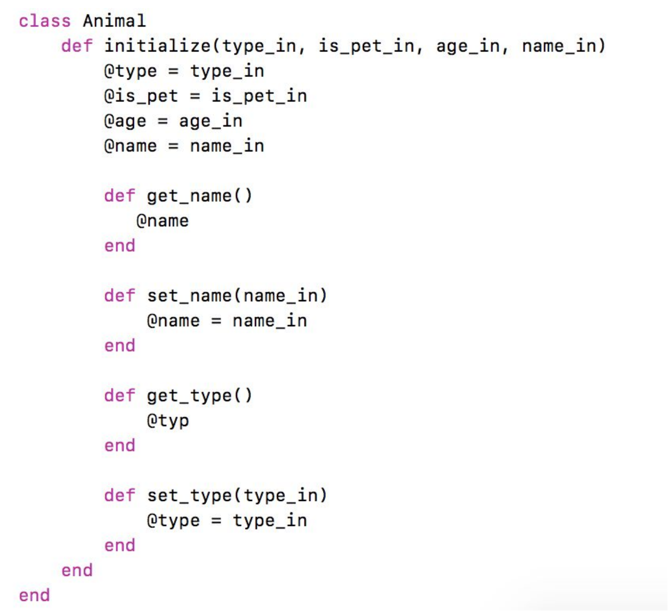
#5.

```
class Example
    @@class_info = ""
    def initialize
        @makes_sense = false
    end
    def set_class_info
        @@class_info = "A class can have class variables, instance variables, and methods. The class itself is an object and objects that are instances of that class can also be
            created."
    end
    def learn
        @makes_sense = true
    end
end
example_1 = Example.new()
example_1.set_class_info()
example_1.learn()
```

#6.

```
class Parent
    attr_accessor :human
    attr_accessor :happy
    @@human = true
    def initialize
       @happy = true
    end
    def set_happy(is_happy = false) #Encapsulation - Controlled access to variable from outside of class
        @happy = is_happy
    end
end
class Child < Parent #Inheritance - Child is an instance of Parent
end
sam = Child.new()
puts sam.happy #returns true
sam.set_happy(false)
puts sam.happy #returns false
```

#7.



```
class Animal
    def initialize(type_in, is_pet_in, age_in, name_in)
        @type = type_in
        @is_pet = is_pet_in
        @age = age_in
        @name = name_in

        def get_name()
           @name
        end

        def set_name(name_in)
            @name = name_in
        end

        def get_type()
            @typ
        end

        def set_type(type_in)
            @type = type_in
        end
    end
end
```

#8. Attr_reader creates a getter method for the variable, attr_writer creates a setter method, and attr_accessor creates both in one statement.

<u>Thinking Assignment</u>

In order to most efficiently find the fluctuation point **n**, we should take the size of the array and divide it by two to get the middle point. We then take the number at the middle point and determine if the next number in the array is greater than or less than that number. If the number to the right of the middle is larger, then we can eliminate the first half of the array as we know the fluctuation does not occur there if the numbers are still increasing in the second half of the array. If the number to the right of the middle point is smaller than the middle number, we can eliminate the second half of the array as we know the fluctuation point has already occured in the first half of the array. We can repeat this process of finding the middle of the array, evaluating it, and then halving the scope of the search until we find a middle number where the number to the right of it is lesser and the number to the left of it is greater.