Austin Osborn
aeo0015@auburn.edu
Project 1 – Ruby

Part 1: Short Answer

1. puts “Hello World”

2. In Ruby, Array and Hashes are natively supported. This includes the basis lists, stacks, queues, trees, dictionaries, etc. However, since Ruby is an object-oriented language, it can theoretically support all types of data structures.

3.
```ruby
#Create array of 1 to 10 and print it to console
array = Array(1..10)
puts array
```

Output:
```
1
2
3
4
5
6
7
8
9
10
```

4.
```ruby
#Create a dictionary of my personal info
dictionary = {
    "first_name" => "Austin",
    "last_name" => "Osborn",
    "tiger_email_id" => "aeo0015",
    "banner_id" => 903682049,
    "favorite_movies" => ["Miracle","Mr. Nobody","Pulp Fiction"]
}
puts dictionary
```

Output:
```
{"first_name"=>"Austin", "last_name"=>"Osborn", "tiger_email_id"=>"aeo0015",
"banner_id"=>903682049, "favorite_movies"=>["Miracle", "Mr. Nobody", "Pulp Fiction"]}
```

5. A class is a blueprint to create a particular type of object. It defines the variables and methods of the class. It models a real world concept. An object is a particular instance of the class (created through an initializer of the class) that is generated and stored in memory to be used in the actual execution of the program.

Example of a class would be the Printer class modeled in the tutorial session:

```
class Printer
  attr_accessor :ink_level
  attr_accessor :page
  def initialize(ink_level, page_amt)
    @ink_level = ink_level
    @page = page_amt
  end
  #Instance Method
  def print(page_count)
    if @ink_level > 0 && @page >= page_count
      #Do some printing
      for i in 1..page_count
        puts "Print pages"
      end
    else
      puts "Not Enough pages or ink"
    end
  end
  #Class Method
  def self.myClassMethod
  end
end
```

Example of an object:
E

```
#Create an object with identifier “printer”
printer = Printer(100, 250)
#Perform a method with the object
printer.print(10)
```

6. Inheritance is the principle of a specific class being a subtype of a more generic “parent” class. Inheritance allows the “subclass” to keep all of the attributes (methods and variables) of the “parent” class and add more/overwrite as necessary. Encapsulation is the concept of the bundling of data with the methods that operate on that data. It can be used to hide the values or state of structured data inside a class, preventing other pieces of code from having direct access to them. Abstraction is the concept of ruducing something into its essential characteristics. This both simplifies code and can create structures that are highly reusable. An example of this would be the abstract data types, which allow users to inherit them for the sake of a specific implementation.

```ruby
#Inheritence/Encapsulation/Abstraction Example
class Animal
  attr_accessor :species
  attr_accessor :age
  def initialize(species_input, age_input)
    @species = species_input
    @age = age_input
  end
end
# Dog inherits from animal
class Dog < Animal
  attr_accessor :name
  def initialize(species, age, name)
    super(species, age)
    @name = name
  end
  def getAge ()
    return self.age
  end
  def setSpecies (species)
    @species = species
  end
  #Dogs go woof, in addition to being an animal
  def woof(volume)
    if volume > 5
      puts "WOOF"
    else
      puts "woof"
    end
  end
end
dog = Dog.new("Great Dane", 7, "Tony")
puts dog.name #prints Tony
puts dog.getAge #prints 7
dog.setSpecies("Shih Tzu")
puts dog.species #Now prints "Shih Tzu" instead of "Great Dane"
```

7.The above code also satisfies the answer to #7 as well.

8. attr_reader is a method that only allows the user to get an attribute of an object.
   attr_writer is a method that only allows the users to set an attribute of an object.
   attr_accessor allows both getting and setting an attribute of an object

Part 4:Thinking Assignment

To find the inflection point index, we must find the point where the previous values are less and the next values are less that the current value. We scan do a scan on the list and compare the current value with the next value. At the point where the next value is less than the current value, this is the inflection index that we return. This will be of O(n) complexity. We can do even better than this by performing a binary search on the array and checking if the left or right side of the split have already inflected. We keep track of the start and end indexes with respect to the original array and select the side that hasn’t been inflected yet. We will arrive at the answer this way with O(lgn) complexity, making it a more efficient solution than that of the naive approach of the linear scan.