Homework Problems:

Problem 1:

```
# Sam Zandiasadabadi
# CSC 600-01
# HW 3.2: Ruby
# 05/18/2024
# Problem 1: Ruby demo program that illustrates the use of all main Ruby iterators

# writing a Ruby demo program for the 'loop' iterator
puts "loop:"
num = 0
loop do
        # displaying the current value of 'num' in the iteration
        puts num
        # increasing the value of num by 1
        num += 1
        # once the value of num equals 4, we end this demo
        if num == 4
        break
        end
end

# writing a Ruby demo program for the 'while' iterator
puts "\nwhile:"
num = 1
while num <= 4
        # while the value of nume is less than or equal to 4, we print
        # the value of num within the current iteration
        puts num
        # increasing the value of num by 1
        num += 1
end

# writing a Ruby demo program for the 'until' iterator
puts "\nuntil:"
num = 2
# executing the loop 'until' the value of the iteration is bigger than or equal to 6
until num >= 6
        puts num
        num += 1
end
```

```
# writing a Ruby demo program for the 'for' iterator
puts "\nfor:"
# for every value in between numbers 3 and 6, we display the values
for num in 3..6 do
        puts num
end

# writing a Ruby demo program for the 'upto' iterator
puts "\nupto:"
# starting from number 4, and iterating upto number 7. we display all values
4.upto(7) do |num|
        puts num
end

# writing a Ruby demo program for the 'downto' iterator
puts "\ndownto:"
# starting from number 7, and iterating downto number 4. we display all values
7.downto(4) do |num|
        puts num
end

# writing a Ruby demo program for the 'times' iterator
puts "\ntimes:"
# running the loop until the value gets to 4
4.times do |num|
        puts num
end

# writing a Ruby demo program for the 'each' iterator
puts "\neach:"
num = 0
# writing sample labels to test the each iterator
numText = ['Number 0', 'Number 1', 'Number 2', 'Number 3', 'Number 4']
numText.each do |numText|
        # for every iteration in the array, we display the current values
        puts "Label: #{num} -> #{numText}"
        num += 1
end

# writing a Ruby demo program for the 'map' iterator
puts "\nmap:"
```

```
# defining a short array, and then adding 1 to each value in the array
puts [1, 2, 4, 6, 8].map{|num| num + 1}

# writing a Ruby demo program for the 'step' iterator
puts "\nstep:"
# starting from number 0, we iterate by 3 values until we reach number 15.
(0..15).step(3) do|num|
        puts num
end

# writing a Ruby demo program for the 'collect' iterator
puts "\ncollect:"
# defining a sample array
numArr = [1, 2, 4, 6, 8]
# using the 'collect' iterator to collect the values from the array and display them
collectValue = numArr.collect{|num| (num)}
puts collectValue

# writing a Ruby demo program for the 'select' and 'reject' iterator
puts "\nselect:"
# creating an array
numArr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
# selecting and displaying all values smaller than 5
puts numArr.select {|num| num < 5}
puts "\nreject:"
# rejecting all values smaller than 5 and displaying the remaining ones
puts numArr.reject {|num| num < 5}
```

Problem 2:

```
        # Sam Zandiasadabadi
        # CSC 600-01
        # HW 3.2: Ruby
        # 05/18/2024
        # Problem 2: Ruby recognizer methods limited? and sorted? that expand the Ruby
class Array.
        # creating a class Array
        class Array
                # defining the 'limited?' method
                def limited?(amin,amax)
                # iterating through each 'num' element in the array that this function is called
                self.each do |num|
                # returning 'true' as long as 'num' is smaller than the largerst index index
        # value, and bigger than the smallest value
        return true unless amax >= num && amin >= num
        end
        # returning false if other wise
        return false
        end

        # defining the "sorted?" method
        def sorted?
        # initializing the value to 0
        indexValue = 0
        # iterating through each 'num' element in the array that this function is called
        self.each do |num|
        # if 'num' is less than or equal to the next element, and we have not reached
        # the end of the array, we continue looping.
        # However, if it is bigger, then we break the loop.
        break unless num <= self[indexValue + 1] if indexValue != self.length - 1
        # returning "+1" if the sequence is increasing
        return "+1" if indexValue == self.length - 1
        indexValue += 1
        end

        # resetting the value back to 0
        indexValue = 0
        self.each do |num|
        # the exact opposite logic to the previous statement above
        break unless num >= self[indexValue + 1] if indexValue != self.length - 1
        # returning "-1" if the sequence is decreasing
        return "-1" if indexValue == self.length - 1
        indexValue += 1
        end

        # if the array is not sorted, then we return 0
        return 0

        end

end


# example 1 of this method
numArray = [1, 2, 4, 6, 8, 10]
```

```
    puts "array: #{numArray}\narray.limited?: #{numArray.limited?(1, 10)}\narray.sorted?: #{numArray.sorted?}"

    # example 2 of this method
    numArray = [10, 8, 6, 4, 2, 1]
    puts "\narray: #{numArray}\narray.limited?: #{numArray.limited?(10, 1)}\narray.sorted?: #{numArray.sorted?}"

    # example 3 of this method
    numArray = [10, 1, 8, 2, 6, 4]
    puts "\narray: #{numArray}\narray.limited?: #{numArray.limited?(10, 4)}\narray.sorted?: #{numArray.sorted?}"
```

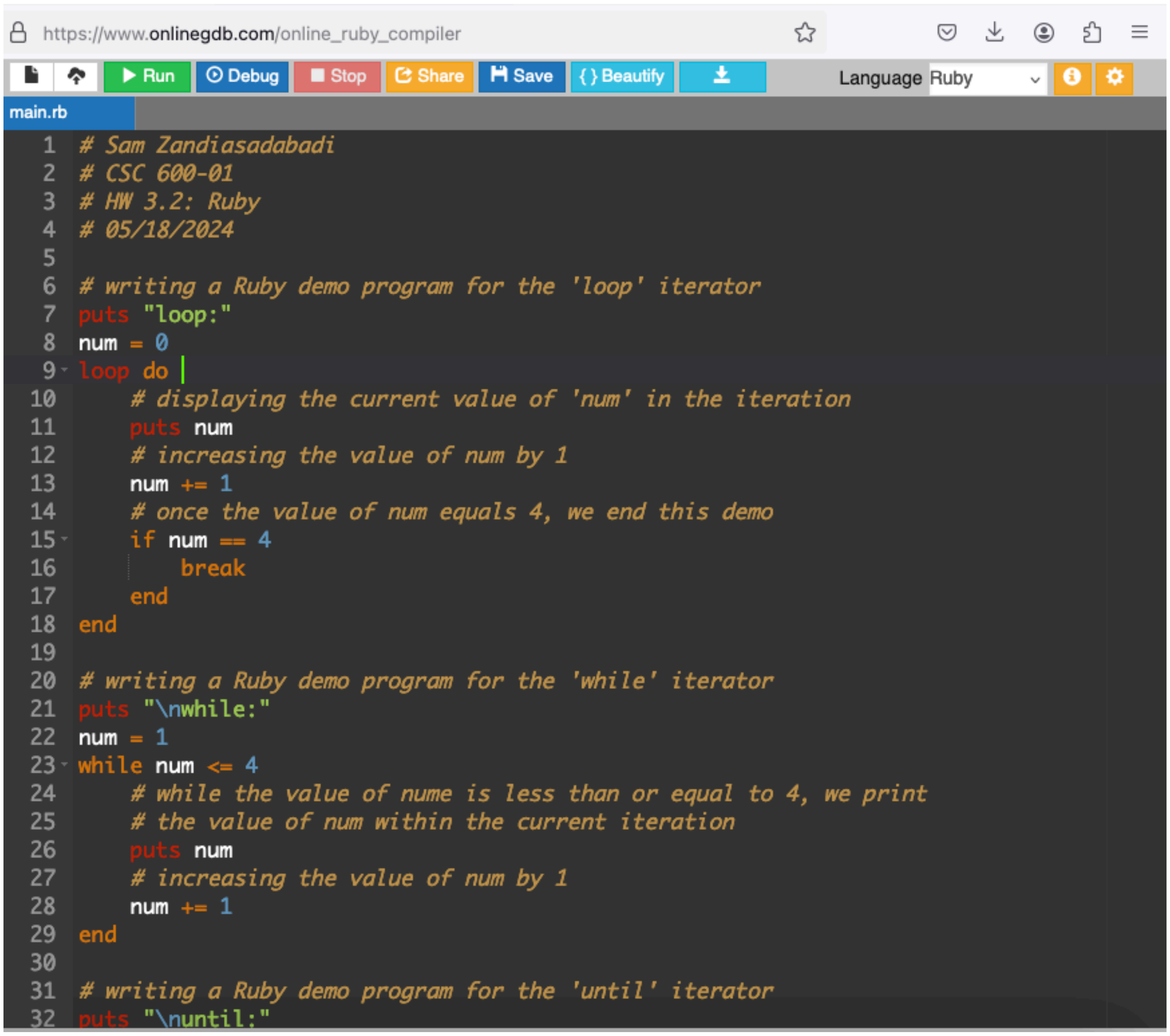
Problem 4:

```
    # Sam Zandiasadabadi
    # CSC 600-01
    # HW 3.2: Ruby
    # 05/18/2024
    # Problem 4: Ruby class Sphere. Each sphere is characterized by the instance variable radius.
    #       Ruby class Ball. Inherits properties from the class Sphere and adds a new instance variable color.
    #       Ruby class MyBall. Inherits properties from the class Ball and adds a new instance variable owner


    # creating the Sphere class
    class Sphere
        # class constructor function
        def initialize(radius)
        @radius = radius
        end

        # creating this public function that returns the value of the radius
        def radius
        @radius
        end

        # defining a function that calculates the area of the sphere
        def area
        # using the formula given in the instructions
        4 * (@radius ** 2) * Math::PI
        end

        # defining a function that calculates the volume of the sphere
        def volume
        # using the formula given in the instructions
        4 * (@radius ** 3) * Math::PI / 3.0
        end

        # defining the show method that displays the instance variables of the class
        def show
        puts "Radius: #{self.radius}"
        end

        # creating a method that displays the area and volume of the sphere class
        def areaAndVolume
        puts "Area: #{self.area}"
        puts "Volume: #{self.volume}"
        end

end

# creating the Ball class by inheriting from the Sphere class
class Ball < Sphere
        # class constructor function
        def initialize(radius, color)
        @radius = radius
        # adding the color variable
        @color = color
        end

        # creating this public function that returns the value of the color
        def color
        @color
        end

        # defining the show method that displays the instance variables of the class
        def show
        puts "Radius: #{self.radius}"
        puts "Color: #{self.color}"
        end

end

# creating the MyBall class that inherits from the Ball class
class MyBall < Ball
        # class constructor function
        def initialize(radius, color, owner)
        @radius = radius
        @color = color
        # adding the owner variable
        @owner = owner
        end

        # creating this public function that returns the value of the owner
        def owner
        @owner
        end

        # defining the show method that displays the instance variables of the class
        def show
        puts "Radius: #{self.radius}"
        puts "Color: #{self.color}"
        puts "Owner: #{self.owner}"
        end

end

# creating a new Sphere object with radius size 2
sphere = Sphere.new(2)
puts "Sphere created"
# displaying the size of the radius
sphere.show
# displaying the area and the volume of the Sphere object
```

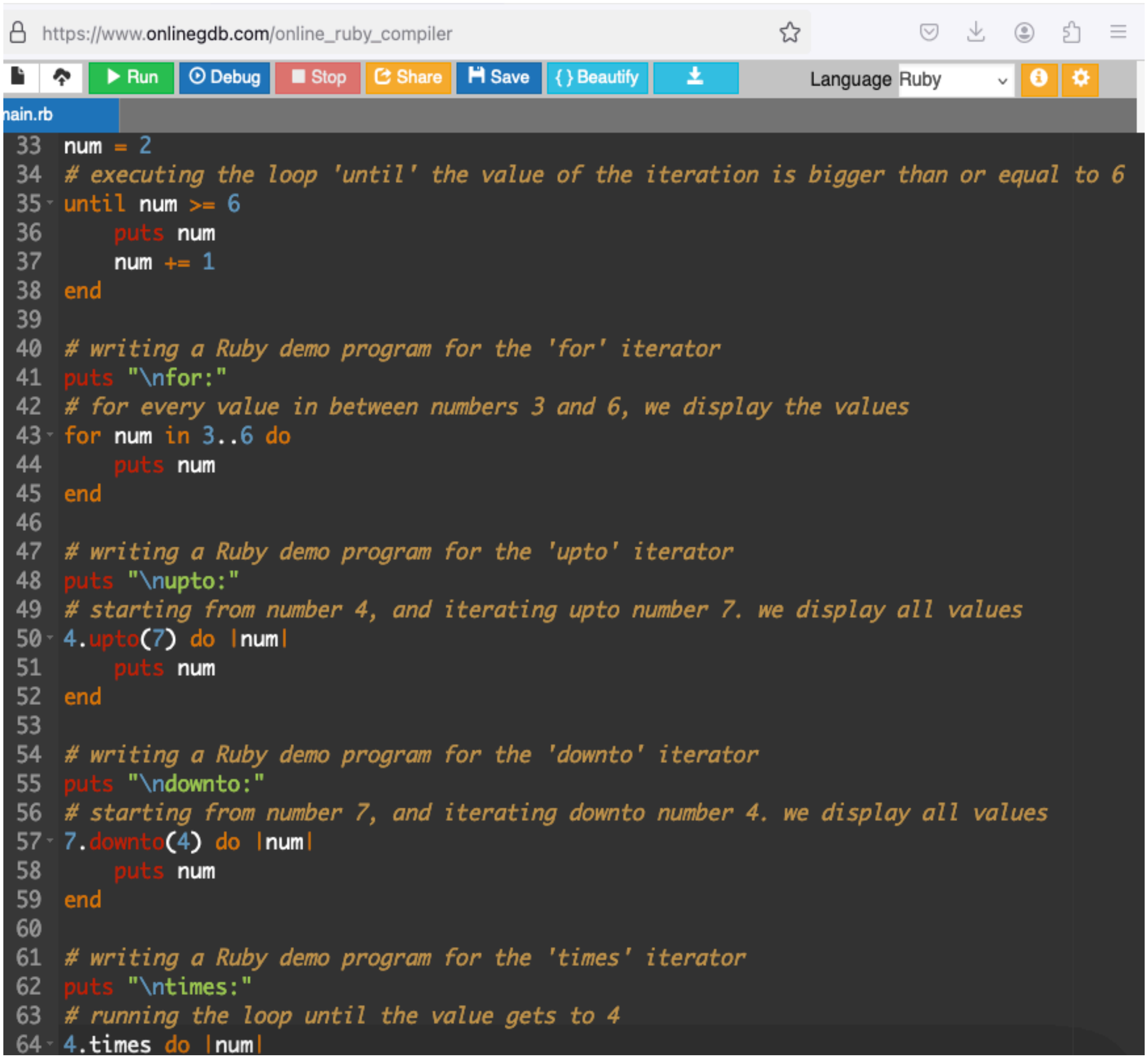
```
sphere.areaAndVolume

# creating a new Ball object with radius size 4 and color black
ball = Ball.new(4, "black")
puts "\nBall created"
# displaying the size of the radius
ball.show
# displaying the area and the volume of the Ball object
ball.areaAndVolume

# creating a new MyBall object with radius size 6, color green, and owner Sam
myball = MyBall.new(6, "green", "Sam")
puts "\nMyBall created"
# displaying the size of the radius
myball.show
# displaying the area and the volume of the MyBall object
myball.areaAndVolume
```

Screenshot of my Code:

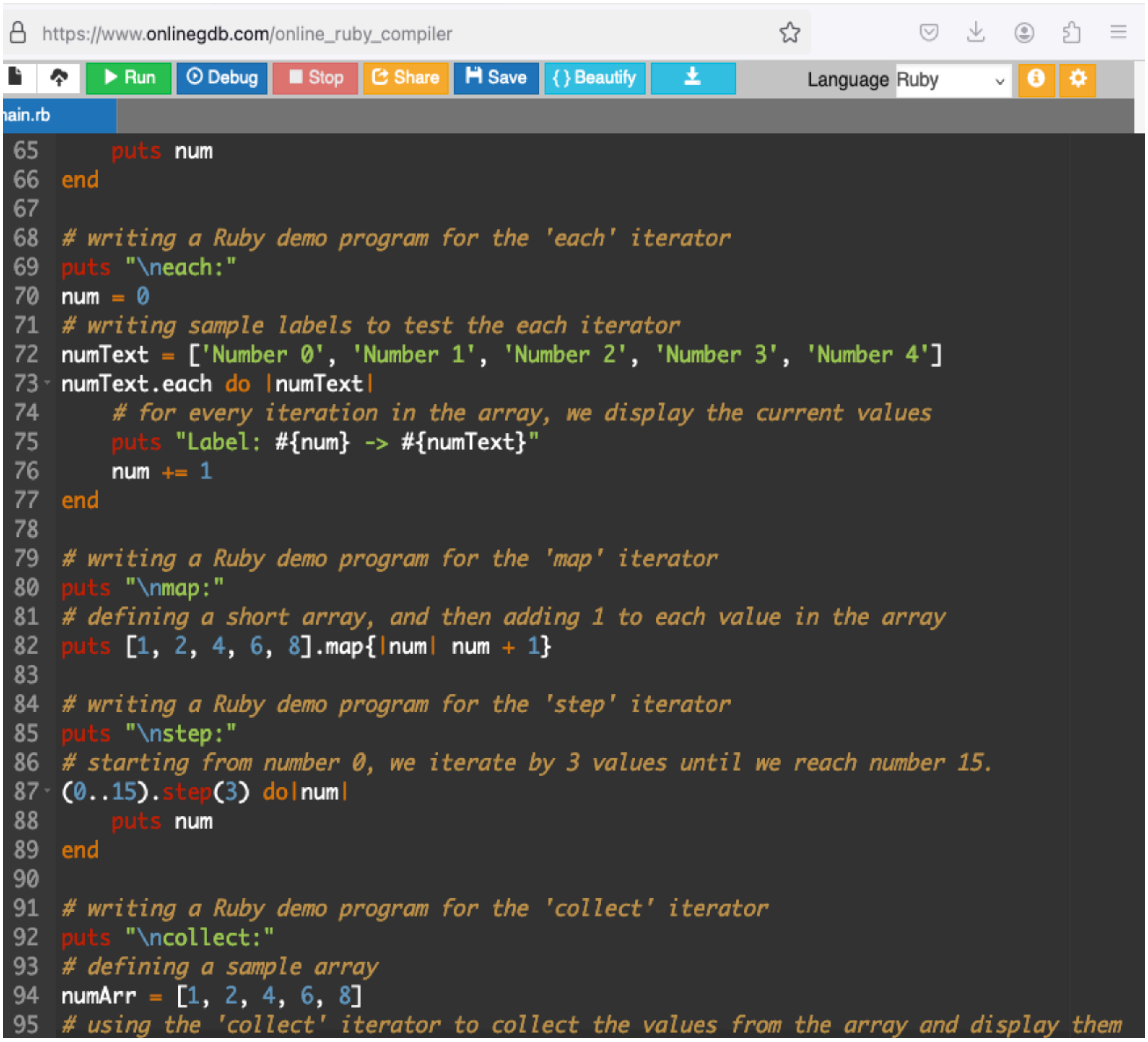
Problem 1:



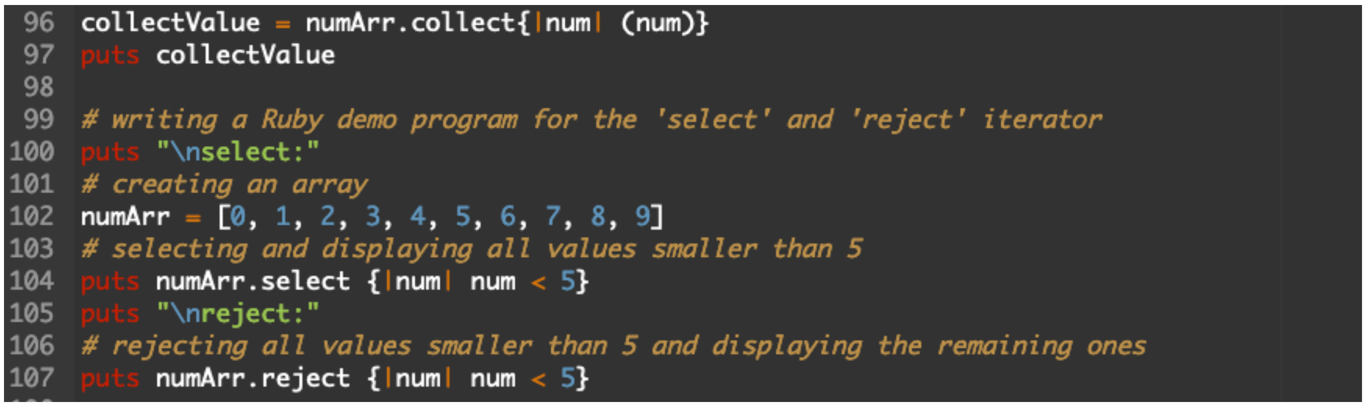
```ruby
# Sam Zandiasadabadi
# CSC 600-01
# HW 3.2: Ruby
# 05/18/2024

# writing a Ruby demo program for the 'loop' iterator
puts "loop:"
num = 0
loop do
    # displaying the current value of 'num' in the iteration
    puts num
    # increasing the value of num by 1
    num += 1
    # once the value of num equals 4, we end this demo
    if num == 4
        break
    end
end

# writing a Ruby demo program for the 'while' iterator
puts "\nwhile:"
num = 1
while num <= 4
    # while the value of nume is less than or equal to 4, we print
    # the value of num within the current iteration
    puts num
    # increasing the value of num by 1
    num += 1
end

# writing a Ruby demo program for the 'until' iterator
puts "\nuntil:"
```



```ruby
num = 2
# executing the loop 'until' the value of the iteration is bigger than or equal to 6
until num >= 6
    puts num
    num += 1
end

# writing a Ruby demo program for the 'for' iterator
puts "\nfor:"
# for every value in between numbers 3 and 6, we display the values
for num in 3..6 do
    puts num
end

# writing a Ruby demo program for the 'upto' iterator
puts "\nupto:"
# starting from number 4, and iterating upto number 7. we display all values
4.upto(7) do |num|
    puts num
end

# writing a Ruby demo program for the 'downto' iterator
puts "\ndownto:"
# starting from number 7, and iterating downto number 4. we display all values
7.downto(4) do |num|
    puts num
end

# writing a Ruby demo program for the 'times' iterator
puts "\ntimes:"
# running the loop until the value gets to 4
4.times do |num|
```

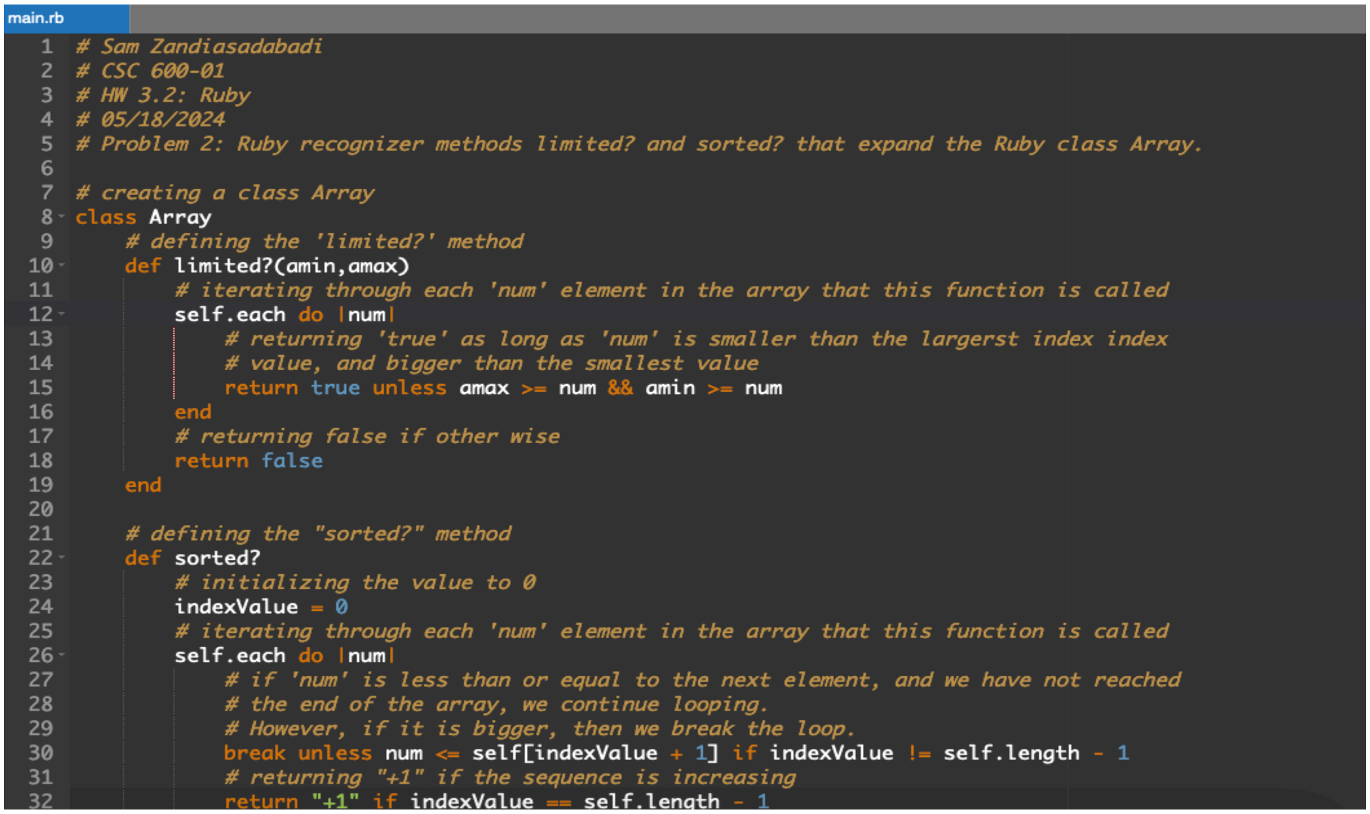


```ruby
    puts num
end

# writing a Ruby demo program for the 'each' iterator
puts "\neach:"
num = 0
# writing sample labels to test the each iterator
numText = ['Number 0', 'Number 1', 'Number 2', 'Number 3', 'Number 4']
numText.each do |numText|
    # for every iteration in the array, we display the current values
    puts "Label: #{num} -> #{numText}"
    num += 1
end

# writing a Ruby demo program for the 'map' iterator
puts "\nmap:"
# defining a short array, and then adding 1 to each value in the array
puts [1, 2, 4, 6, 8].map{|num| num + 1}

# writing a Ruby demo program for the 'step' iterator
puts "\nstep:"
# starting from number 0, we iterate by 3 values until we reach number 15.
(0..15).step(3) do|num|
    puts num
end

# writing a Ruby demo program for the 'collect' iterator
puts "\ncollect:"
# defining a sample array
numArr = [1, 2, 4, 6, 8]
# using the 'collect' iterator to collect the values from the array and display them
```



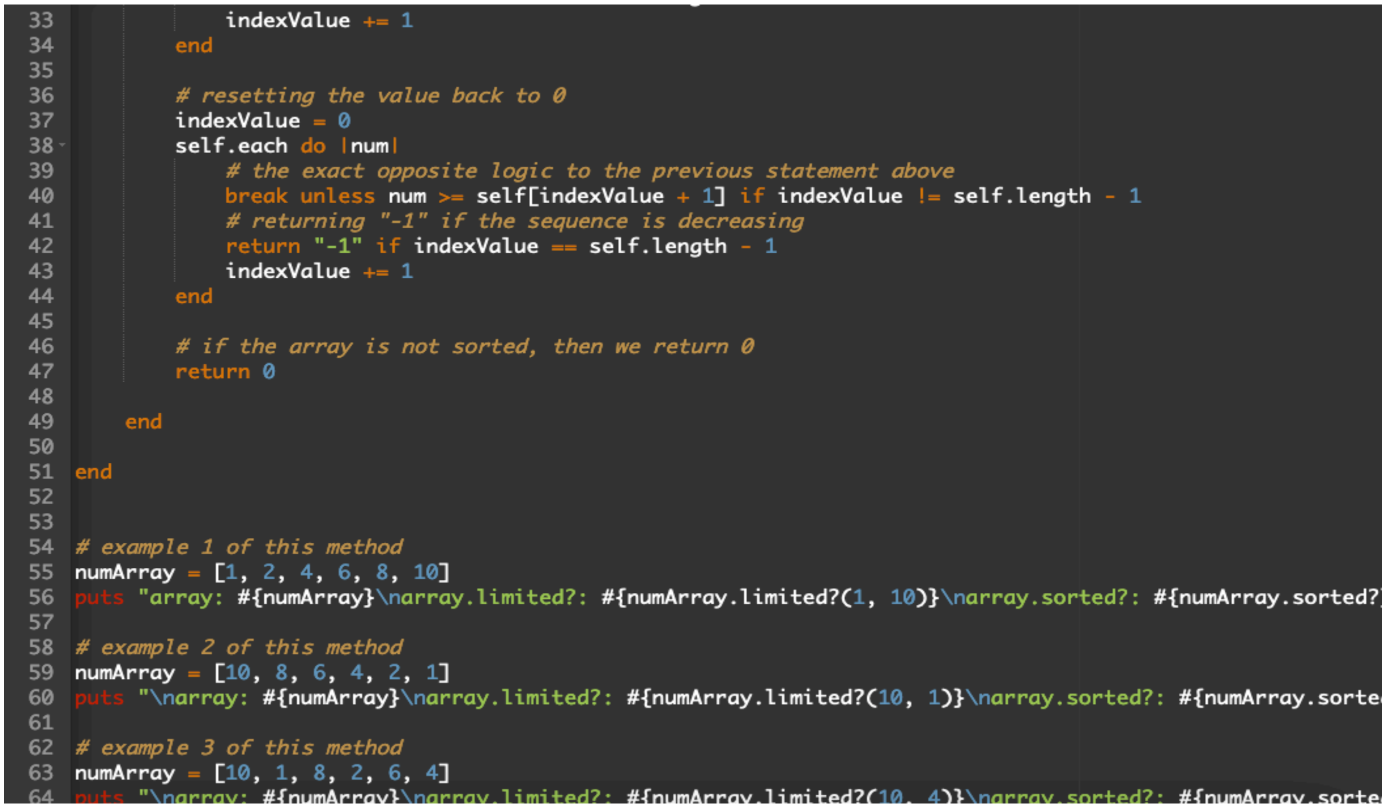
```ruby
collectValue = numArr.collect{|num| (num)}
puts collectValue

# writing a Ruby demo program for the 'select' and 'reject' iterator
puts "\nselect:"
# creating an array
numArr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
# selecting and displaying all values smaller than 5
puts numArr.select {|num| num < 5}
puts "\nreject:"
# rejecting all values smaller than 5 and displaying the remaining ones
puts numArr.reject {|num| num < 5}
```

Problem 2:



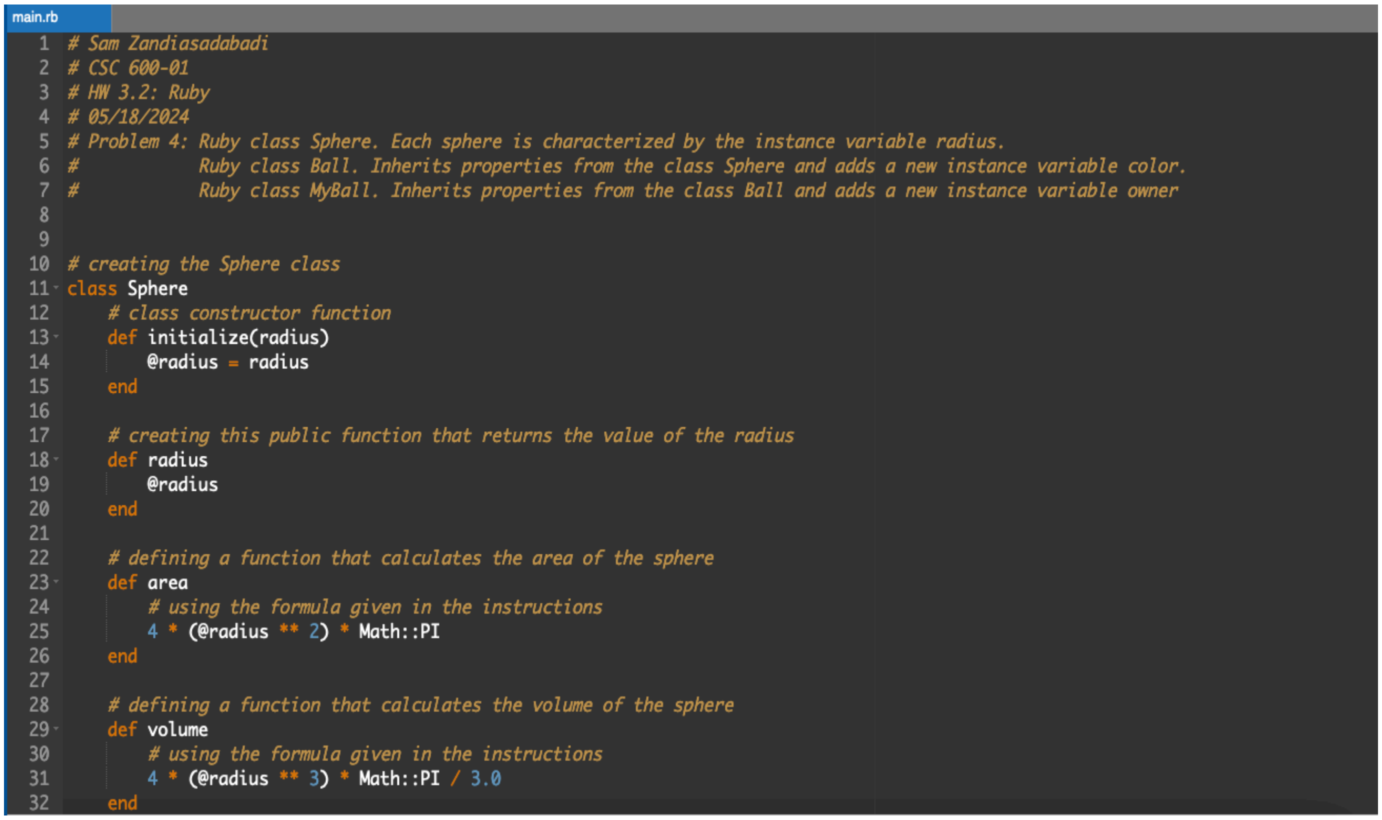
main.rb

```ruby
# Sam Zandiasadabadi
# CSC 600-01
# HW 3.2: Ruby
# 05/18/2024
# Problem 2: Ruby recognizer methods limited? and sorted? that expand the Ruby class Array.

# creating a class Array
class Array
    # defining the 'limited?' method
    def limited?(amin,amax)
        # iterating through each 'num' element in the array that this function is called
        self.each do |num|
            # returning 'true' as long as 'num' is smaller than the largerst index index
            # value, and bigger than the smallest value
            return true unless amax >= num && amin >= num
        end
        # returning false if other wise
        return false
    end

    # defining the "sorted?" method
    def sorted?
        # initializing the value to 0
        indexValue = 0
        # iterating through each 'num' element in the array that this function is called
        self.each do |num|
            # if 'num' is less than or equal to the next element, and we have not reached
            # the end of the array, we continue looping.
            # However, if it is bigger, then we break the loop.
            break unless num <= self[indexValue + 1] if indexValue != self.length - 1
            # returning "+1" if the sequence is increasing
            return "+1" if indexValue == self.length - 1
```



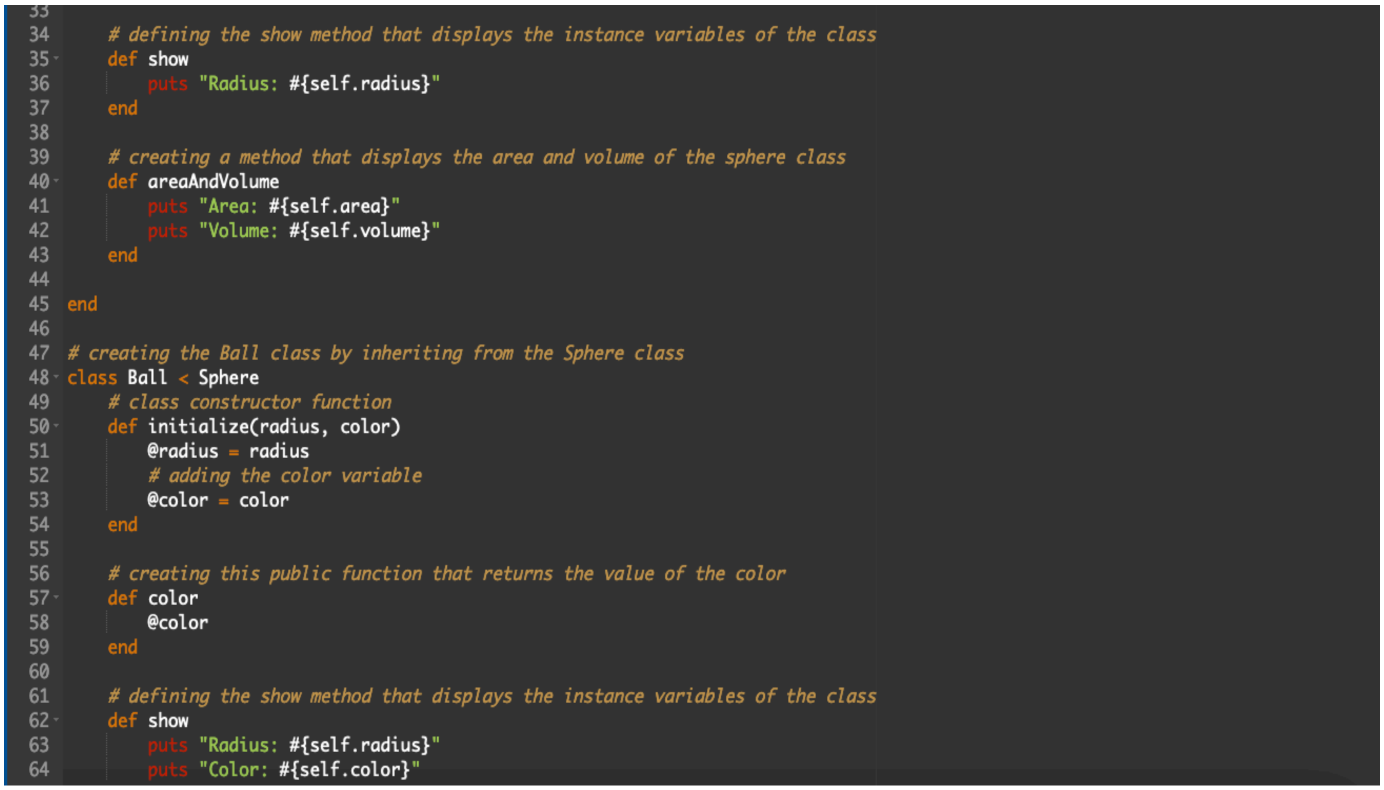
```ruby
            indexValue += 1
        end

        # resetting the value back to 0
        indexValue = 0
        self.each do |num|
            # the exact opposite logic to the previous statement above
            break unless num >= self[indexValue + 1] if indexValue != self.length - 1
            # returning "-1" if the sequence is decreasing
            return "-1" if indexValue == self.length - 1
            indexValue += 1
        end

        # if the array is not sorted, then we return 0
        return 0

    end

end


# example 1 of this method
numArray = [1, 2, 4, 6, 8, 10]
puts "array: #{numArray}\narray.limited?: #{numArray.limited?(1, 10)}\narray.sorted?: #{numArray.sorted?

# example 2 of this method
numArray = [10, 8, 6, 4, 2, 1]
puts "\narray: #{numArray}\narray.limited?: #{numArray.limited?(10, 1)}\narray.sorted?: #{numArray.sorte

# example 3 of this method
numArray = [10, 1, 8, 2, 6, 4]
puts "\narray: #{numArray}\narray.limited?: #{numArray.limited?(10, 4)}\narray.sorted?: #{numArray.sorte
```

Problem 4:

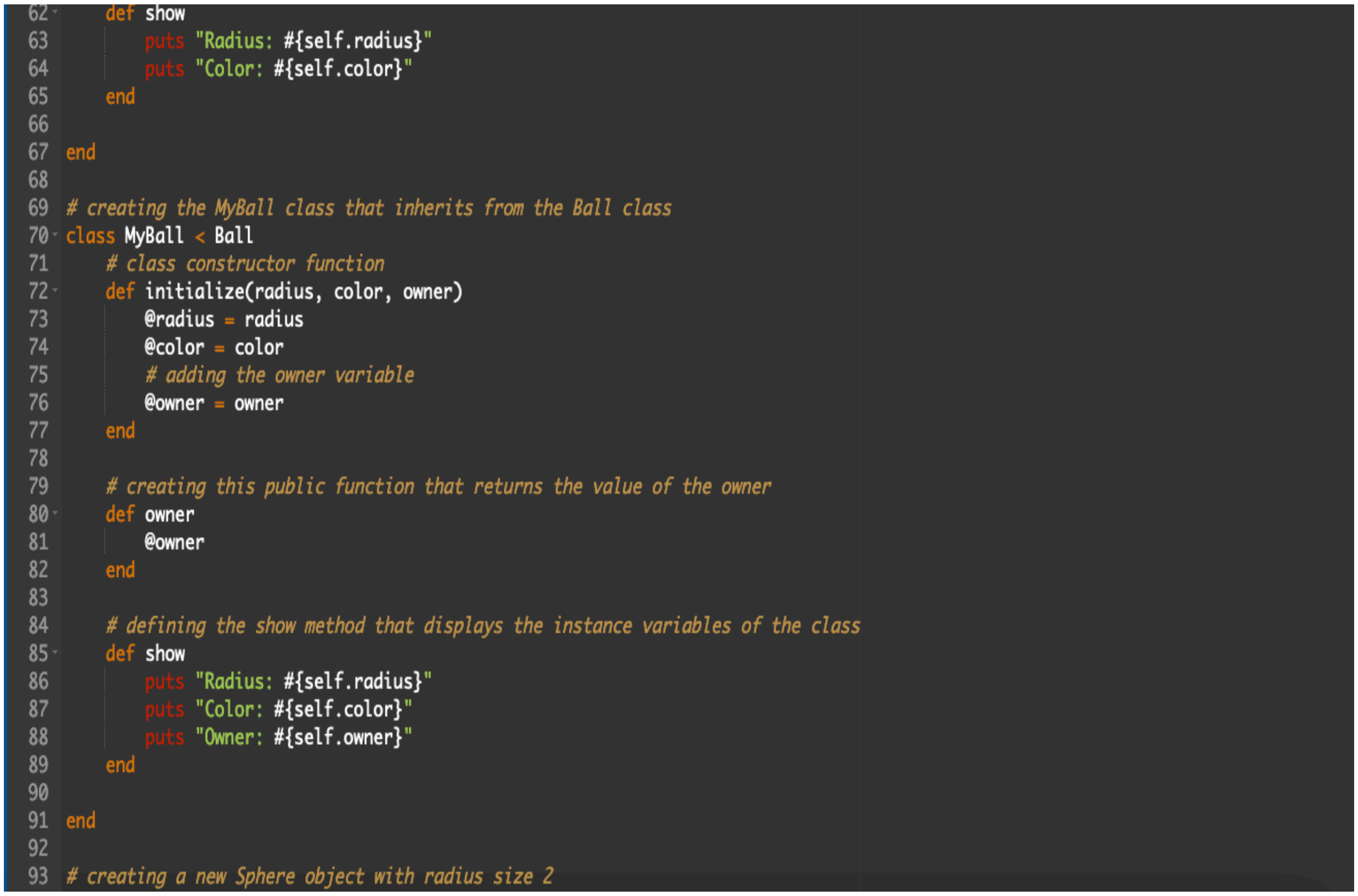


main.rb

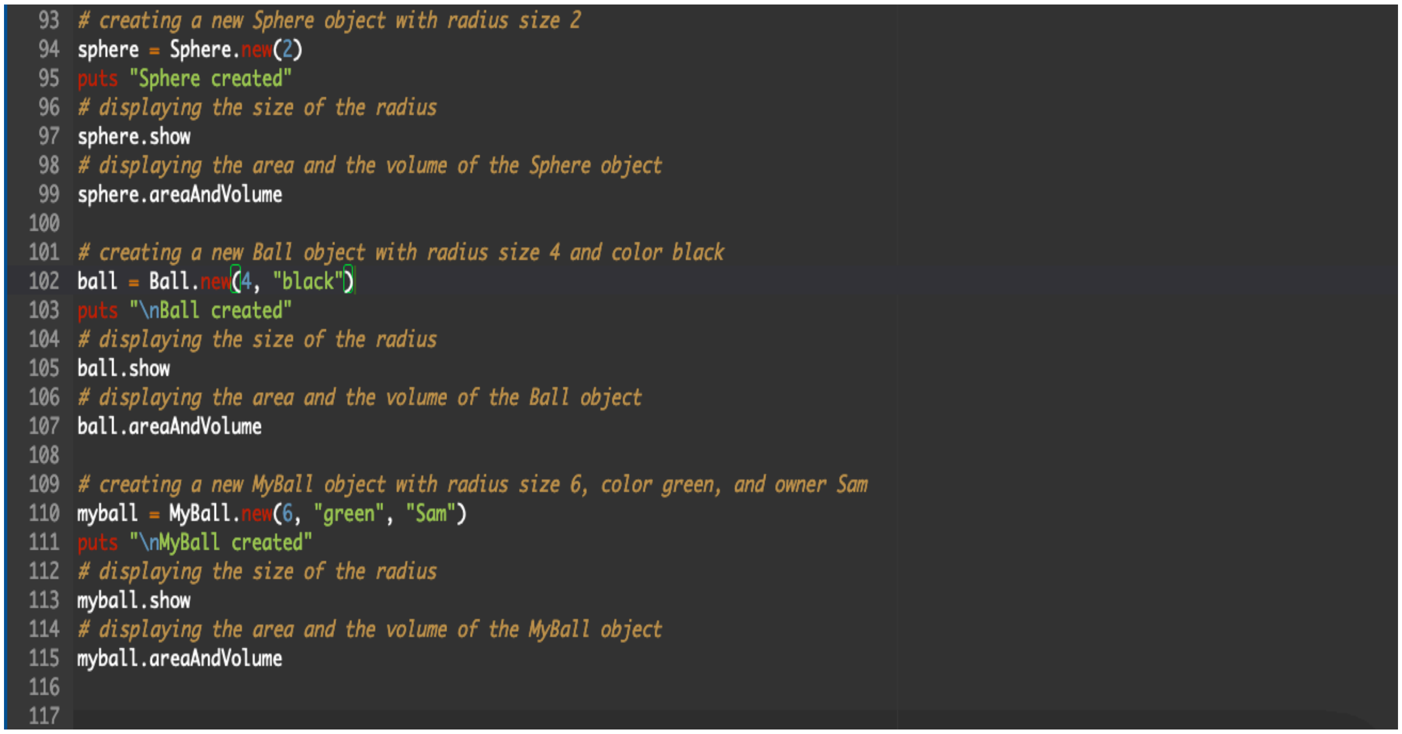
```ruby
# Sam Zandiasadabadi
# CSC 600-01
# HW 3.2: Ruby
# 05/18/2024
# Problem 4: Ruby class Sphere. Each sphere is characterized by the instance variable radius.
#            Ruby class Ball. Inherits properties from the class Sphere and adds a new instance variable color.
#            Ruby class MyBall. Inherits properties from the class Ball and adds a new instance variable owner


# creating the Sphere class
class Sphere
    # class constructor function
    def initialize(radius)
        @radius = radius
    end

    # creating this public function that returns the value of the radius
    def radius
        @radius
    end

    # defining a function that calculates the area of the sphere
    def area
        # using the formula given in the instructions
        4 * (@radius ** 2) * Math::PI
    end

    # defining a function that calculates the volume of the sphere
    def volume
        # using the formula given in the instructions
        4 * (@radius ** 3) * Math::PI / 3.0
    end
```



```ruby

    # defining the show method that displays the instance variables of the class
    def show
        puts "Radius: #{self.radius}"
    end

    # creating a method that displays the area and volume of the sphere class
    def areaAndVolume
        puts "Area: #{self.area}"
        puts "Volume: #{self.volume}"
    end

end

# creating the Ball class by inheriting from the Sphere class
class Ball < Sphere
    # class constructor function
    def initialize(radius, color)
        @radius = radius
        # adding the color variable
        @color = color
    end

    # creating this public function that returns the value of the color
    def color
        @color
    end

    # defining the show method that displays the instance variables of the class
    def show
        puts "Radius: #{self.radius}"
        puts "Color: #{self.color}"
```



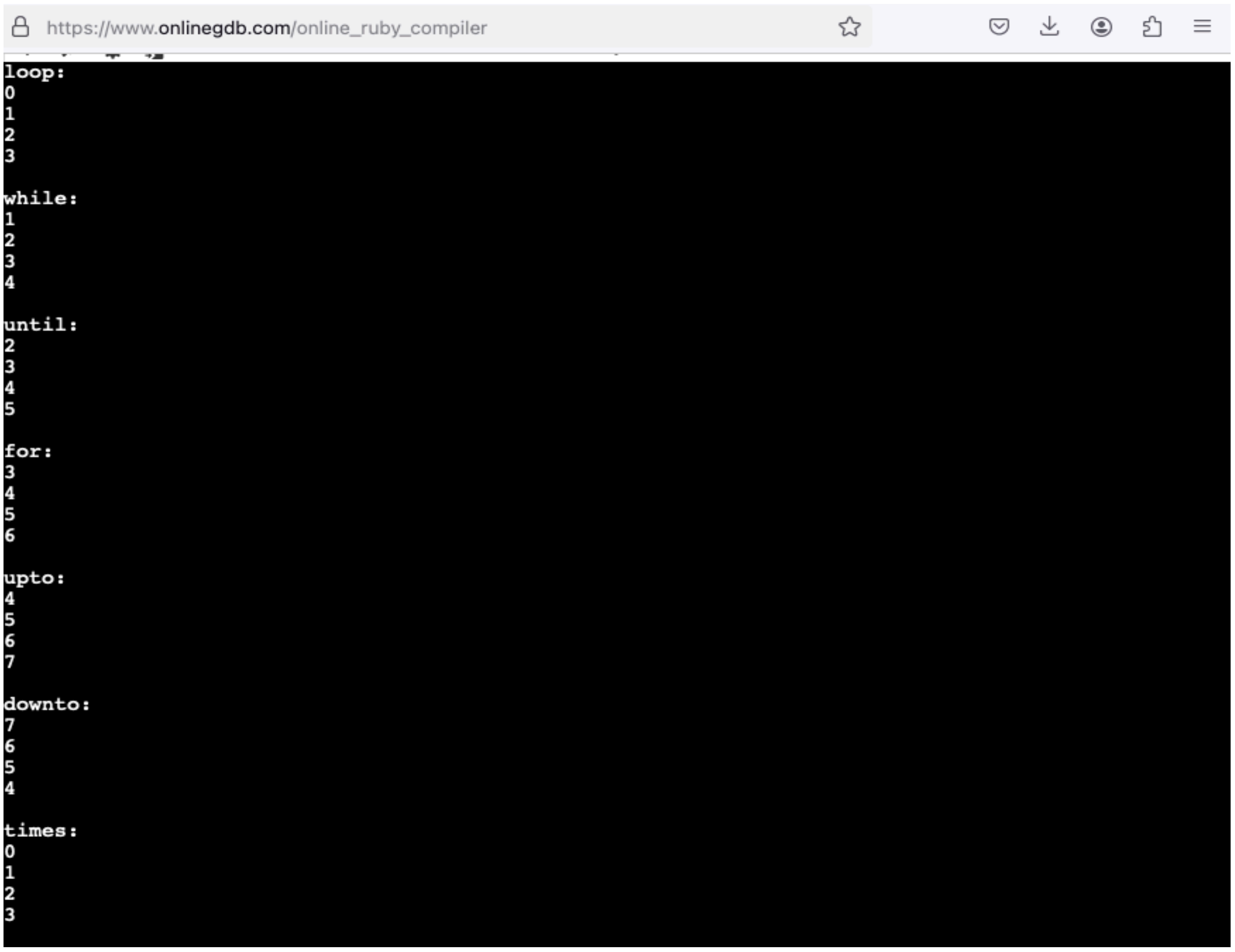
```ruby
    def show
        puts "Radius: #{self.radius}"
        puts "Color: #{self.color}"
    end

end

# creating the MyBall class that inherits from the Ball class
class MyBall < Ball
    # class constructor function
    def initialize(radius, color, owner)
        @radius = radius
        @color = color
        # adding the owner variable
        @owner = owner
    end

    # creating this public function that returns the value of the owner
    def owner
        @owner
    end

    # defining the show method that displays the instance variables of the class
    def show
        puts "Radius: #{self.radius}"
        puts "Color: #{self.color}"
        puts "Owner: #{self.owner}"
    end

end

# creating a new Sphere object with radius size 2
```



```ruby
# creating a new Sphere object with radius size 2
sphere = Sphere.new(2)
puts "Sphere created"
# displaying the size of the radius
sphere.show
# displaying the area and the volume of the Sphere object
sphere.areaAndVolume

# creating a new Ball object with radius size 4 and color black
ball = Ball.new(4, "black")
puts "\nBall created"
# displaying the size of the radius
ball.show
# displaying the area and the volume of the Ball object
ball.areaAndVolume

# creating a new MyBall object with radius size 6, color green, and owner Sam
myball = MyBall.new(6, "green", "Sam")
puts "\nMyBall created"
# displaying the size of the radius
myball.show
# displaying the area and the volume of the MyBall object
myball.areaAndVolume

```

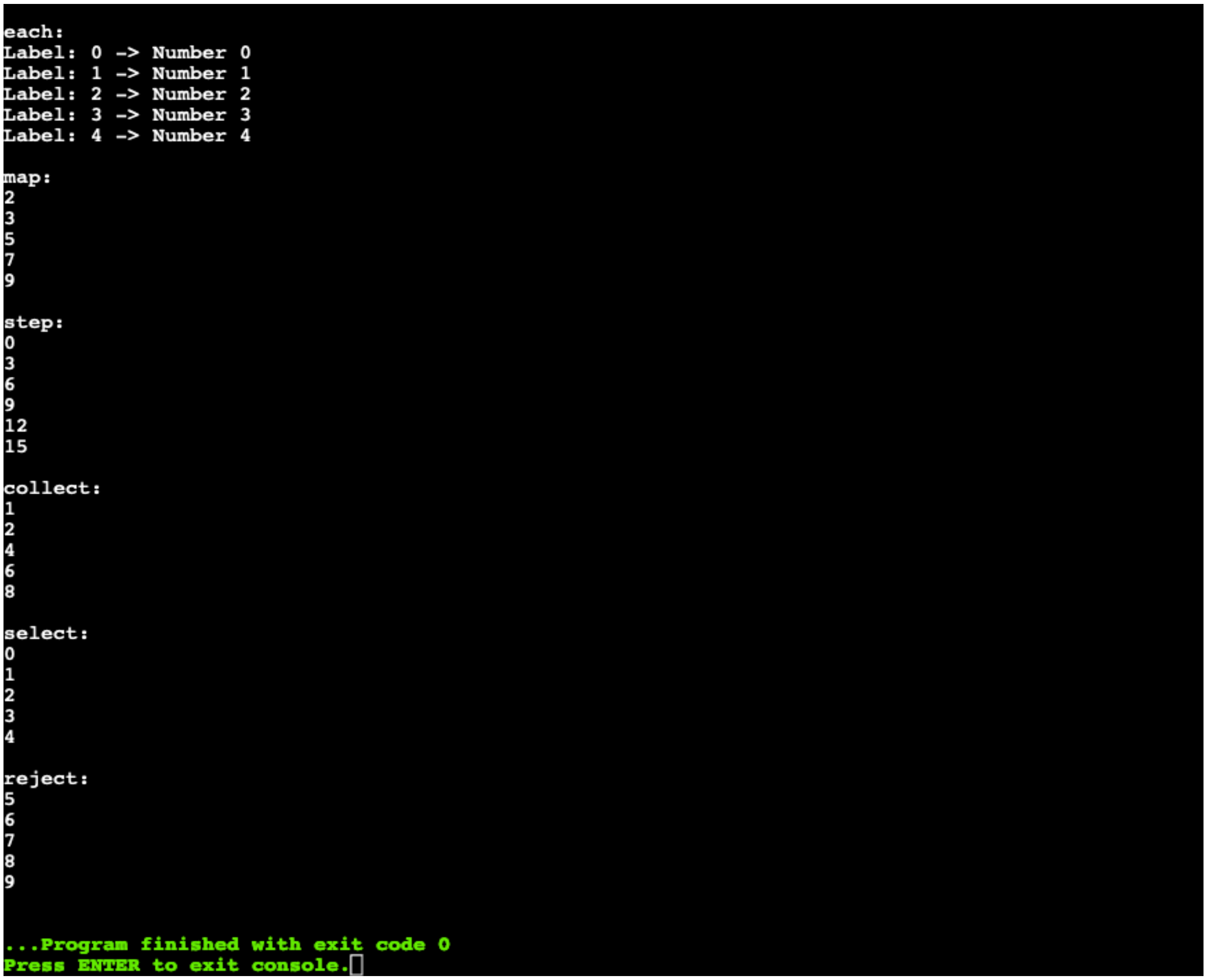
Output:

Problem 1:



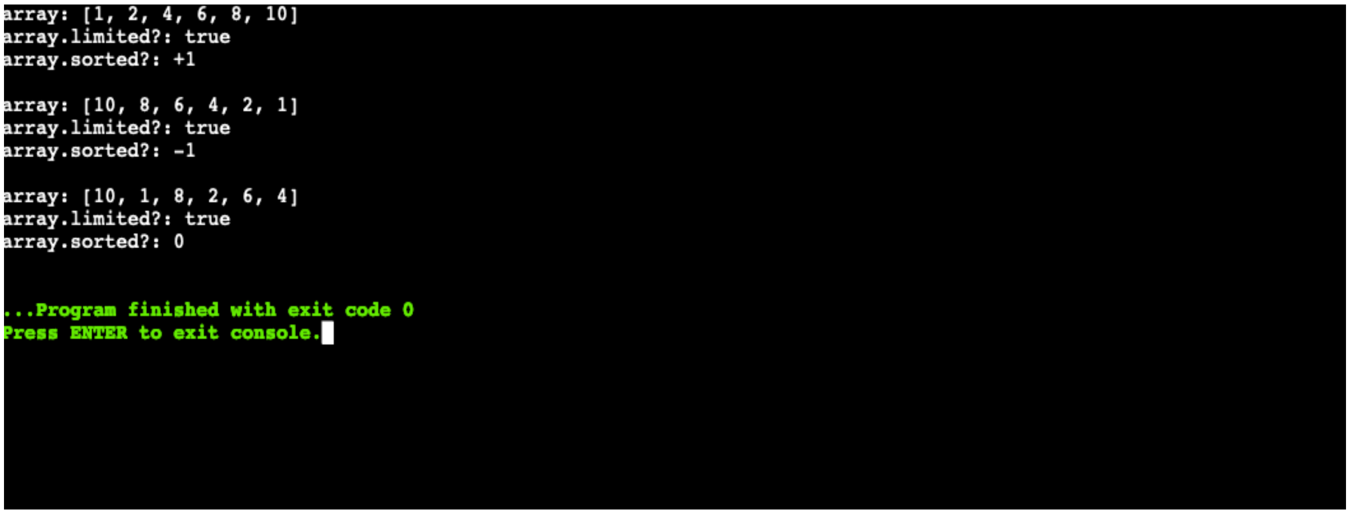
https://www.onlinegdb.com/online_ruby_compiler

```
loop:
0
1
2
3

while:
1
2
3
4

until:
2
3
4
5

for:
3
4
5
6

upto:
4
5
6
7

downto:
7
6
5
4

times:
0
1
2
3
```



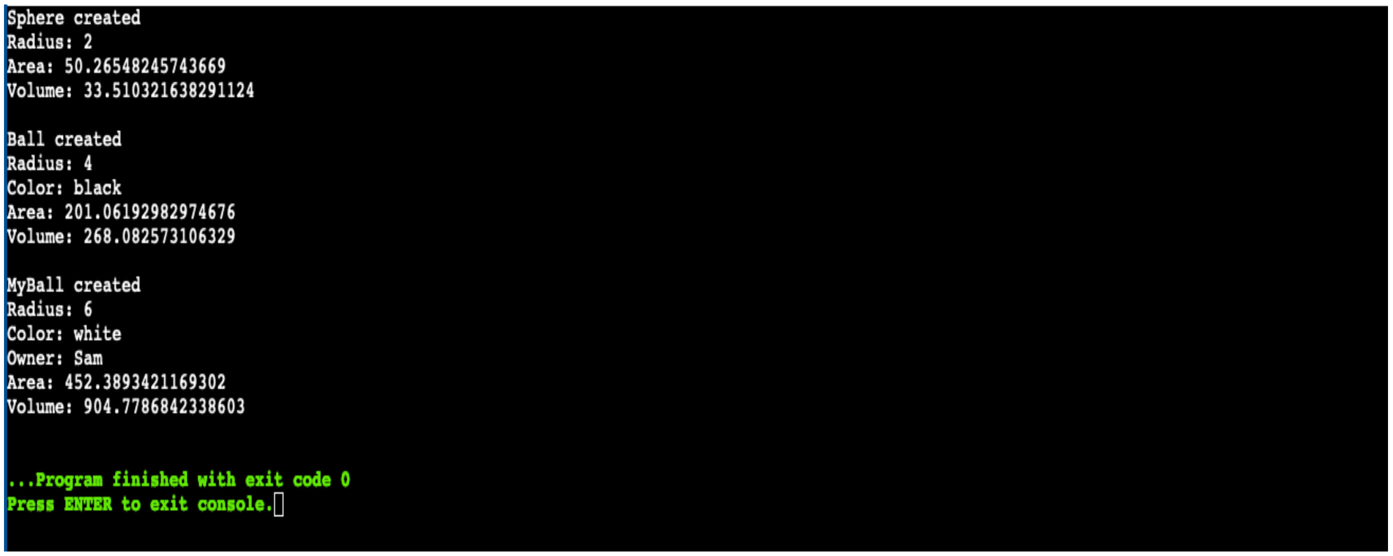
```
each:
Label: 0 -> Number 0
Label: 1 -> Number 1
Label: 2 -> Number 2
Label: 3 -> Number 3
Label: 4 -> Number 4

map:
2
3
5
7
9

step:
0
3
6
9
12
15

collect:
1
2
4
6
8

select:
0
1
2
3
4

reject:
5
6
7
8
9


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 2:



```
array: [1, 2, 4, 6, 8, 10]
array.limited?: true
array.sorted?: +1

array: [10, 8, 6, 4, 2, 1]
array.limited?: true
array.sorted?: -1

array: [10, 1, 8, 2, 6, 4]
array.limited?: true
array.sorted?: 0


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 4:



```
Sphere created
Radius: 2
Area: 50.26548245743669
Volume: 33.510321638291124

Ball created
Radius: 4
Color: black
Area: 201.06192982974676
Volume: 268.082573106329

MyBall created
Radius: 6
Color: white
Owner: Sam
Area: 452.3893421169302
Volume: 904.7786842338603


...Program finished with exit code 0
Press ENTER to exit console.
```