**Group-based Activity 2; CS 3060**

**Names** of students in your group: Nicholas W, Dylan S, Ethan F, Nicholas B, Tyson S

Points: 10 (with 2 bonus points)

**Task 1**: (4 points) Write a Ruby function foo which takes an array (whose name is *arr*) of integers (say i1, i2, …). In the body of foo, use the *map* (or *collect*) function to compute the remainder of each integer x (e.g., i1, i2, …) when x is divided by 3. Function foo should return the array of remainders.

Test your code, and include the test results in a readme file..

Hint: We discussed about *map* construct in the previous class. If you forgot, first you may test the following code (just as practice) on the *irb* console. Guess what it is doing.

*arr = [“def”, “xy”, “wsd”]*

*arr.map{|item| item.size}*

# Task 1

def foo(*arr*)

    return arr.map{ |x| x = x%3}

end

puts foo([12, 15, 18, 65, 74])

**Task 2**: (4 points) Implement a function *genRandomMatrix* to generate a matrix. The function *genRandomMatrix* should take one parameter n and return a matrix of size n x n. Use the given template below. Generate a variety of random matrices that are filled with random integers between 10 and 20. You should test your code for matrix sizes of 3x3 and 15x15. For visualization, you must also create one helper function *printMatrix*. The *printMatrix(A)* will print matrix A in a well-formatted fashion.

def genRandomMatrix(*n*)

    a = [ ]

    (1..n).each do

            b = [ ]

            (1..n).each do

                    i = rand(10..20) #you may need to fill in a parameter in rand( ) here

                    b.push(i)

            end

            a.push(b)

    end

    a

end

def printMatrix(*a*)

    a.each do |row|

        print row

        print "\n"

    end

end

randMatrix = genRandomMatrix(3)

printMatrix(randMatrix)

Now extend your code to generate a rectangular matrix. Then, run your code to generate and print a 10x15 matrix. Include the test results in a readme file.

**Task 3**: (4 points) Study the following code. This is about using *code block*.

def seriesUpTo(*max*) # this func receives a code block

    i = 1

    while i <= max

        yield i\*i    # see correspondence b/w i\*i and x in block later.

        i += 1

    end

end

seriesUpTo(100) { |x| puts x + 1 } # this will print some series

puts "\n"

Answer these questions: (a) how many times is the *code block* (in blue) executed in the above? Explain your answer. (b) Edit the above code to print this series (i.e., only values): 12 + 1, 22 + 1, 32 + 1, …, 1002 + 1

Submission: Upload one copy of your work (this word doc with completed code and a separate readme file with results) per group to Canvas.