# Long Exam: COM201 and INF201

Fundamentals of Programming

September 22, 2020

***Instruction.*** The total score of the long exam that will be recorded is **150 points**[1]. You may choose the questions that you want to answer to accumulate scores for the exam. Note that each question has corresponding grading system. Write your pseudocode/flowchart in Word file and convert to it PDF file.

***How to submit?*** Archieve (i.e. Zip) your Java files and PDF files. Then, submit the Zip file to **mvoctavianojr@national-u.edu.ph** with email subject: "**Long Exam: <Section>**". Indicate in your email body the members of your group.

## 1 Pre-game

**A**. Convert the Equation 1 into Java codes and display the value of V. Ask the user for input values of the variable(s) if needed.

$$V = (\frac{9q^{9/3}}{3q^3})/(3 + (\frac{4s^{-8/2}}{4} + 2)(\frac{2^{6/2}}{8})) \tag{1}$$

**Grading:**

- Code correctness: 10 points
- Coding style: 5 points

**B**. Create a program that displays the sum of all the multiples of 3 or 5 but not by 6 below 1000. For example, 12 must not be added because it is multiples of 6.

**Grading:**

- Code correctness: 10 points
- Pseudocode: 5 points

[1]This will be split into 3 records: Long exam 1, 50 points; Long exam 2, 50 points; Long exam 3, 50 points.

**C**. Create a program that asks input integers M and N. The program must display the sum of 10 odd-values in a Fibonnaci sequence where M and N are the starting values. In general, the Fibonacci sequence are series of numbers where the value of $F_n$ is based from the sum of the previous numbers $F_{n-1}$ and $F_{n-2}$. For example,

M = 20
N = 21
Fibonacci sequence: 20, 21, 41, 62, ...

*NOTE:* The M and N are excluded on the count of odd-values to be added.

**Grading:**

- Code correctness: 10 points
- Pseudocode: 5 points

## 2 Minor League

**A**. Create a program that accepts an array $A$ with *1* x $n$,

$$A = \begin{bmatrix} a_1, & a_2, & \dots & a_n \end{bmatrix}$$

and displays the output $V$ where,

$$V = \sum_{k=1}^{n} \prod_{x=k}^{n} a_x a_k + a_x$$

**Grading:**

- Code correctness: 15 points
- Flow chart: 5 points

**B**. Create a method *isPalindromic(A): boolean* that accepts a string value and returns True when A is palindrome when characters are rotated. Otherwise, return False. A palindrome is a word that when reads forward, it reads the same on backward.

**Test case 1:**
isPalindromic("mmaad")
*Output:* True

**Test case 2:**
isPalindromic("window")
*Output:* False

**Grading:**

- Code correctness: 10 points
- Flow chart: 10 points

**C**. Create a method *whatArray(A)* that accepts 2-dimensional array with $m$ x $n$ dimension and prints,

$$whatArray(A) = \begin{cases} print\text{“}square'', & \text{if } m = n \\ print\text{“}skinny'', & \text{if } m > n \\ print\text{“}fat'', & \text{if } m < n \\ print\text{“}rectangular'', & \text{if } m \neq n \end{cases}$$

**Grading:**

- Code correctness: 15 points
- Flow chart: 5 points

**D**. Write a Java program that finds a trio of elements (i.e. indices of the three numbers) from a given array *A* whose sum equals a specific target number *B*.

**Test case 1:**
A=[10,20,10,40,50,60,70]
B=40
*Output:* 0, 1, 2

**Test case 2:**
A=[10,20,10,40,50,60,70]
B=80
*Output:* 0, 1, 4

**Grading:**

- Code correctness: 10 points
- Flow chart: 10 points

**E**. Create a method *addArray(A)* that accepts a 1-dimensional array. Elements in the array is considered an integer as whole of the method. The method will add 1 to the whole number representation and returns the corresponding array representation of the sum. For example, the method represents the *A=[1,2,3]* as a whole number *123*. The method will return a new array, *A′=[1,2,4]*.

**Test case 1:**
A=[5,6,7]
*Output:* A′=[5,6,8]

**Test case 2:**
A=[9,9,9]
*Output:* A′=[1,0,0,0]

**Grading:**

- Code correctness: 10 points
- Flow chart: 10 points

## 3 Major League

**A**. Create a method *newArray(A, B)* that accepts 2-dimensional arrays, $A$ and $B$, and returns a new array $C$. If array $A$ has $m$ x $n$ and array $B$ has $n$ x $p$,

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}, B = \begin{bmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{np} \end{bmatrix}$$

The new array C will have an $m$ x $p$ dimension wherein each element of the matrix is defined as:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} + b_{kj}$$

for $i = 1, ..., m$ and $j = 1, ..., p$.

**Grading:**

- Code correctness: 20 points
- Pseudocode: 5 points

**B**. Create a method *reduce(M)* that accepts a 2-dimensional array $M$ that has $n$ x $n$ dimension where n $\geq$ 2, 4, 6, 8, ...,

$$M = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

When the array folds vertically, the method will display "Horray" if all elements on the left part will have the same values. And, the right part will also have the same values that is different on the left part. For example, the method will display "Hooray" on the following arrays:

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}, B = \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \end{bmatrix}$$

**Grading:**

- Code correctness: 20 points
- Pseudocode: 5 points

High scores from the long exam will become worthless if you learn nothing.
Resist the temptation to copy and paste the solution of others.
Goodluck!