MAD 2502 Assignment 1 (Due **Feb 5th Monday, 5:00PM**)

Instructions:

* This is a computer-based assignment (Total points: 8). Please write your code that runs in Python.

(1) Submission: Your codes/programs must be submitted electronically by email to sbai@fau.edu by 5:00 PM on the above due date. Use your **fau.edu email address** to send the code (unless you do not have one). Your email header/subject line should be:

MAD2502-HW1-Name

Include the Python code as an attachment with filename (put all your functions/codes in this single Python file):

MAD2502-HW1-Name.py

(2) Naming: A comment at the top of your program file should also identify yourself and the assignment that you are submitting:

```
# Assignment 1
# Name:
# Z-Number:
```

Please also follow the requirement(s) after each question to *name* your functions.

(3) Z-Number: Some question(s) may depend on your Z-Number thus each student will have a **unique** question and solution. Make sure you use your own Z-Number in your answer.

(4) Python commands/libraries: Do not use Python libraries unless it is mentioned in the question. Do not use any Python pre-defined commands/libraries that have not been discussed in this class. For example, do not use the Python built-in command that computes the sum.

(5) Comments/citation: Please comment your code. All code should be written by you. If any part of what you turn in is not your own work – you learn it from books, webpages etc – the source must be referenced.

MAD 2502 Assignment 1 (Due **Feb 5th Monday, 5:00PM**)

Q1. (1 points) Write a program which **inputs** a non-negative integer n and **prints** the factorial of n, e.g.,

$$n! = 1 \cdot 2 \cdot 3 \cdots n.$$

We also define $0! = 1$. Test your program by computing $n!$ for a few numbers including $n = 0$; $n = 1$ and $n = 73$.

Requirement: write a function (with comments) with name "Factorial",

```
### Solution to Question 1
def Factorial(n):
        ... Your codes ...
```

The function should return the factorial of n computed. One can thus print the result using, e.g.

```
# Assume user inputs n = 73
result = Factorial(n)
print(result)
# should print 4470115461512684340891257138 ... omit ...
```

Do not use recursion (in case you already know that).

Q2. (2 points) Continue with the above question. Write a program that computes and prints out the first m factorials of **ODD** numbers (where the value for m is from user input), **in reverse order**. See the following example for the requirement.

Requirement: define a function with name "AllOddFactorials",

```
### Solution to Question 2
def AllOddFactorials(m):
        ... Your codes ...
```

For example, if $m = 4$, calling "AllOddFactorials(m)" will print 7!, print 5!, print 3! and then print 1! (do not print any factorial of even numbers). The literal outputs should be (for $m = 4$),

```
5040
120
6
1
```

Note the prints must be **in reverse order**. You may re-use part of your code from Question 1.

Q3. (2 points) Continue with above questions (you may use the codes from your above solutions).
First, write a function that computes the following function,

$$f(m) = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{m!}$$

where m is from user input.

Requirement: define a function with name "TotalFactorials" (note this is just above $f(m)$),

```
### Solution to Question 3 (part1)
def TotalFactorials(m):
        ... Your codes ...
```

The function should return the $f(m)$ computed. One can thus print the result by, e.g.

```
# Assume user inputs m = 2
result = TotalFactorials(m)
print(result)
# should print 2.5 here
```

Second, write another function that computes-and-prints $f(m)$ for **every** m starting from $m = 0$ to $m = M$ where M is an user input.

Requirement: define a function with name "AllTotalFactorials" (note this is just above $f(m)$),

```
### Solution to Question 3 (part2)
def AllTotalFactorials(M):
        ... Your codes ...
```

The function should print the $f(0), f(1), f(2), \cdots, f(M)$ consecutively, e.g.

```
# Assume user inputs M = 3
AllTotalFactorials(m)
# should print the following
1
2
2.5
2.6666666666667
```

Q4. (3 points) This question uses your FAU Z-number (should be 8-digits). Please follow the instructions below to generate the question. Denote your Z-number by z.

- If your Z-number starts with 0, truncate the leading consecutive zeros, e.g. 00020202 becomes 20202. Then $z = 20202$.
- If your Z-number does not start with 0, keep all of it. E.g. $z = 22222222$.

In either case, you have a number z now (note this number will be unique for each of the student). **This number will be used later to generate a function.**

First, load the *random* module in your code (e.g. put the following line on top of your program),

```
import random
```

Initialize the random seed using your z and then generate a random number t.

```
### Assume z = 22222222 (your number z will be different)
random.seed(22222222)
# Use 1e4 and 1e5 in the following line. Do not change them.
t = random.randint(1e4, 1e5)
# here a random integer t=45680 is generated
```

Second, defined a function using the above generated t as,

$$f(x) = x^2 + t^{2/3}x + t.$$

Note in Python you use the floating-point number for the $t^{2/3}$. This finishes the preparation for the question.

The question aims to approximate the root of $f(x)$ using **TWO** methods that we've discussed on the class.

(Task 1) First, it is known that analytically the **two** roots of a quadratic polynomial of the form $g(x) = ax^2 + bx + c$ can be calculated by,

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}; r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Substitute your t into $b^2 - 4ac$ and then approximate its square root by the **Bisection** method. In the end, approximate the roots r_i of function $f(x)$ using above formula. We denote these approximated roots by $\tilde{r_1}$ and $\tilde{r_2}$ for future use.

Requirement: define a function with name "RootByMethod1" to implement above idea (first approximate the square root by Bisection and then use the analytic formula). It is **required** that the approximated square root (denote Δ) for $b^2 - 4ac$ has to satisfy

$$|\Delta^2 - (b^2 - 4ac)| \leq 0.05.$$

Make it clear in the code by commenting that **which initial interval** the Bisection method uses and **why** they can be used. Your program should print Δ and the $\tilde{r_1}$ and $\tilde{r_2}$ in the end.

(Task 2) From the above procedure, we've already found some approximated roots $\tilde{r_1}$ and $\tilde{r_2}$ (they are probably coarse approximations). The purpose of this task is to further refine these approximations using whatever method discussed on the class (e.g. Exhaustive Search or Bisection).

Requirement: define a function with name "RefineRoot" to implement above idea. It is **required** that the refined roots $\tilde{\tilde{r_1}}$ and $\tilde{\tilde{r_2}}$ satisfy,

$$|f(\tilde{\tilde{r_i}})| \leq 0.0000001.$$

Note the approximated root $\tilde{r_1}$ and $\tilde{r_2}$ from Task 1 should be inputs for your function. Your program should print the refined approximated roots $\tilde{\tilde{r_i}}$ and errors $|f(\tilde{\tilde{r_i}})|$ in the end.