Algorithm Design

LAB 10 : NUMERICAL SEQUENCE PROGRAM

Due Saturday at 5:00 PM MST

This week, we will implement in Python a design we created previously.

Program Description

This program will prompt the user for a positive integer *n* and then display the *n*th François number.

Asserts

Identify several places in the program which could be the source of an error or which could indicate a potential bug. Place asserts in these locations.

Example

The following examples show two run-throughs of the program. In the first one, we will compute the ninth François number.

```
Which Francois number would you like to see? 9
Francois number 9 is 47.
```

In the second example, we will look at a much bigger number.

```
Which Francois number would you like to see? 100
Francois number 100 is 489526700523968661124.
```

Assignment

To submit this assignment, two things are needed: your source code and a demonstration video.

Source Code

Submit your source code as a file attachment in I-Learn. At the top of your program, include a comment answering these five questions:

```
# 1. Name:
#      -your name-
# 2. Assignment Name:
#      Lab 10: Numeric Sequence
# 3. Assignment Description:
#      -describe what this program is meant to do-
# 4. What was the hardest part? Be as specific as possible.
#      -a paragraph or two about how the assignment went for you-
# 5. How long did it take for you to complete the assignment?
#      -total time in hours including reading the assignment and submitting the program-
```

Demonstration Video

Record a short video demonstrating the execution of your program. The video must be no longer than one minute. Your demonstration video must cover the following test cases:

1. -1
2. 0
3. 1
4. 2
5. 9
6. 100
7. 200

As the video is recorded, mention the test case you are covering.

Assessment

Your grade for this activity will be according to the following rubric:

	Exceptional 100%	Good 90%	Acceptable 70%	Developing 50%	Missing 0%
Code Quality 30%	Perfection! The code is extremely easy to understand and very straightforward	Professional and efficient	A few obvious mistakes were made	Readable	Little effort was spent on style. The code looks thrown together or is missing
Asserts 20%	The summation of the asserts is likely to catch many likely bugs	Two good asserts exist in the design	One assert is sufficiently defined and correctly placed	An assert exists in the program, but it is unlikely to catch a bug	No asserts are mentioned in the design
Functionality 40%	All the test cases execute perfectly	Everything works but there are minor cosmetic defects	One test case fails to execute as expected	At least one test case works as expected	Code does not run, is missing, or does not resemble a working solution
Reflection 10%	The reflection component of the video completely and concisely describes what went well and what went poorly	It is clear that thought went into the reflection component of the video	Each reflection question is addressed, but there is no evidence of introspection	At least one reflection question is answered	There is no voice-over in the video or the voice-over fails to address any of the required points