Florida Institute of Technology
Department of Computer and Engineering Sciences
CSE1100 – Intro to Programming Concepts with Python
Homework Problem Set #2

For this exercise you will perform the following activities:

1. Write software source code in Python that implements the requirements from one of the following problem specifications. Choose the level most appropriate for your amount of programming experience.
    - L1 - This is my first programming class.
    - L2 – I have programmed before for other classes, work, or fun.
    - L3 – I have programmed quite a bit and I'm looking for a challenge.
2. Create and test executable scripts that perform operations as described in the specifications.
3. Complete the standard homework and lab template report to describe your programs including screen shots of test-run results.
4. Submit the complete product including the Python source code as a zip file to the provided Canvas drop box.

IMPORTANT NOTES:

1. Unlike the labs, these HW products will represent substantial individual work. You must include extensive comments that indicate your approach to each solution. Include your name, date, and modification version as comments in the source code as demonstrated in the template examples.
2. Any source code derived from other's work must be indicated. Use the IEEE citation format for web pages and text documents explained here:

   How to Cite References: IEEE Documentation Style

Failure to follow the above guidelines will result in a penalty of no less than 25 percent. If the unattributed work violates the rules of Academic Honesty, as referenced in the syllabus, the assignment will receive 0 (zero) credit.

Problem L1: Fizz – *Buzz* – BANG!!

Like "Hello World," "Fizz Buzz" has become quite famous in the computing world. It is usually given as an example of an "easy" problem.

Try to exercise your creativity when solving this variation of the modern classic. There are many ways to do it. Joel Grus wrote an entire book on it[1] (Grus, 2020).

Specifications:

- Write a program that prints out the first 200 positive integers in 20 lines of 10 comma-separated values with the following edits in-place.
    - If a number is a multiple of 3, replace it with the string "Fizz"
    - If a number is a multiple of 5, replace it with the string "*Buzz*"
    - If a number has both 3 and 5 as factors, replace it with the string "BANG!!"

An example of the first lines would be:

```
1, 2, Fizz, 4, *Buzz*, Fizz, 7, 8, Fizz, *Buzz*
11, Fizz, 13, 14, BANG!!, 16, 17, Fizz, 19, *Buzz*
```

---

[1] Grus, J. (2020) "Ten Essays on Fizz Buzz". Retrieved October 28, 2021, from https://joelgrus.com/2020/06/06/ten-essays-on-fizz-buzz/.

Problem L2: Uniform Dice Simulator

Write a program that simulates the rolling of two n-sided dice. The user will specify the number of sides on each die. An n-sided die is often notated as dn, so a 6-sided die would be written d6 and a 4-sided die would be d4. Summing multiple dice of the same type is noted mdn, where m is the number of dice to be summed. The chance of any side showing up is assumed to be the same as any other (uniform distribution).

The sum of the two values should then be calculated. Note: Each die can show an integer value from 1 to n, so the sum of the values of an n sided die and an m sided die will vary from 2 to $m + n$, with varying frequencies of the intermediate sums. The figure below shows the table for 2d6 (the 36 possible sum combinations of two 6-sided dice).

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Figure 2 - All 36 possible combinations of two d6 dice.

To create the experimental frequency table, your program shall roll the two dice 36,000 times and sum them. During each run of the experiment the program shall:

- Tally the numbers of times each possible sum appears
- Compute the expected vs. actual percentage of sum occurrence
- Print the results in a tabular format as shown in the 2d6 example below.

Possible Sums (d6+d6)	Count	Expected Freq. (%)	Actual Freq. (%)
2	982	2.778	2.7278
3	2063	5.556	5.7306
4	3042	8.333	8.4500
5	3950	11.111	10.9722
6	5015	13.889	13.9306
7	5946	16.667	16.5167
8	4942	13.889	13.7278
9	4071	11.111	11.3083
10	3004	8.333	08.3444
11	1979	5.556	05.4972
12	1006	2.778	2.7944

Hints:

Note that in the 2d6 example, there are 36 possible sum outputs of two dice. Out of the 36, it is expected that the sum of 2 will occur one time, the sum of 3 two times, the sum of 4 three times, and so on. Thus, the expected frequency for the appearance of particular sum, x is $\frac{x}{36}$, where x is the number of ways to achieve that sum on the dice.

Uniformly-distributed random numbers—those that have equal chance of occurrence—can be easily generated in Python using the standard library functions as discussed here: https://docs.python.org/3/library/random.html

NumPy also has an extensive selection of random number generation features as discussed here: https://numpy.org/doc/stable/reference/random/index.html?highlight=random#module-numpy.random

Feel free to use these or another pseudo-random number generation method of your choice.

Problem L3: RPN Calculator

Reverse Polish Notation (RPN) is a way of entering arithmetic expressions into a calculator that obviates the use of parentheses. As opposed to the standard infix notation that is most commonly used, the operators come after their operands in each expression. For example: $2 + 5$ is expressed $2\ 5\ +$, and $6 * 7$ is expressed as $6\ 7\ *$.

In infix notation, expressions such as $2 + 5 \times 6 + 7$ are ambiguous since $7 \times 13 \neq 2 + 30 + 7$. This leads to the need for a defined order-of-operations including parentheses to precisely specify cases that would otherwise be ambiguous.

In RPN, these two calculations are expressed unambiguously as: $2\ 5\ +\ 6\ 7\ + \times$ and $2\ 5\ 6\ \times\ +\ 7\ +$ respectively. These expressions are simply evaluated left to right with operators being evaluated immediately on their operands.

Specifications:

- Write a program that computes and prints the value of RPN expressions.
- The program only needs to support addition, multiplication, subtraction, and division of base-10 numerals.
- Example output would look as follows:

```
Enter an Expression (enter 'Q' to quit): 2 5 + 6 7 + *
Result: 91
Enter an Expression (enter 'Q' to quit): 2 5 6 * + 7 +
Result: 39
```

See The following websites for more on RPN:

https://en.wikipedia.org/wiki/Reverse_Polish_notation

https://mathworld.wolfram.com/ReversePolishNotation.html