General Instructions:

- **Lab is due online by 11:59 pm of the due date.** The assignment must be typed, not handwritten or scanned.

- Label your Python programs `q<num>.py`, where `num` is the question number. For example, your solution to the first question is stored in the file `q1.py`.

- Make sure you read the lab before you start. Your programs must match the output exactly as given for each question. This is important as one of the keys to being a good programmer is attention to details.

- If you forgot to bring a computer, you can check out a temporary laptop from ZACH 383 for a few hours (depending on the availability).

- Grading is based on correctness and clarity. **Copying work from another source and submitting it as your own is plagiarism.** The minimum penalty for plagiarism is a grade of zero.
Lab Questions

1. *Simulation*
   Assuming we use a fair coin, write a Python program stored in a file q1.py that computes, on average, the number of flips necessary until 3 tails up occur in sequence. Your program should execute 1, 10, 100, 1000, 10000, and 100000 experiments, and compute the average on each case.

<table>
<thead>
<tr>
<th>Experiments</th>
<th>Average Flips</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25.00</td>
</tr>
<tr>
<td>10</td>
<td>8.70</td>
</tr>
<tr>
<td>100</td>
<td>12.98</td>
</tr>
<tr>
<td>1000</td>
<td>14.22</td>
</tr>
<tr>
<td>10000</td>
<td>13.98</td>
</tr>
<tr>
<td>100000</td>
<td>14.05</td>
</tr>
</tbody>
</table>

2. *Simulation:
   Answer the following questions in a file q2.txt:
   Suppose you flip a fair coin 3 times independently.
   - What is the probability of getting three tails up in sequence?
   - What is the probability of getting at least one heads up in the sequence?
   - What is the probability of getting at least two tails up in the sequence?
   - What is the probability of getting at most two tails up in the sequence?

3. *Craps: the dice game*
   Write a Python program stored in a file q3.py that plays the dice game called Craps.
   - The player begins by throwing two standard dice.
   - If the sum of the dice is 7 or 11, the player wins.
   - If the sum is 2, 3 or 12, the player loses.
   - Otherwise, the sum becomes the player’s point.
   The player continues to roll until either the point comes up again, in which case the player wins, or the player throws 7, in which case they lose.
**Programming Tips.** Write the program without showing a visual representation of the two dice. Show the two random dice values as integers first. Once your program is working, go back and add the code for converting the dice values into a visual representation.

Finally, when writing the code for representing the dice visually, you might think about taking advantage of common drawing components among the dice. If you can find commonality among how the dice are drawn on the screen, you can reduce the amount of code written to solve the problem. Also, if you use lists in this program to visually represent your dice, you will need to convert the list into a string using the `join` operator.

See the code below.

```python
# A program that converts a list to a string. In order for this
# to work, each element in the list, must be a string.

a_list = ["$", "8884", "yes"]

# Joins list1 elements without a separator.
a_list_string1 = ".join(a_list)

# Joins list1 elements with XYZ as a separator.
a_list_string2 = "XYZ".join(a_list)

print(a_list)
print(a_list_string1)
print(a_list_string2)
```

Note: since random numbers are required, your program will not run in the same manner as the following examples unless the same random numbers are generated.

**Example #1.** After the title of the game is displayed (line 1), the player presses the <Enter> key to roll the two dice (line 3). Then, the visual representation for the dice roll of 5 and 2 is displayed on the screen (lines 3–7). Since a 7 was rolled on the first try, the player wins (lines 8–9).

```
Craps: A Popular Dice Game
Press <Enter> to roll the dice.
+-------+ +-------+
| * * | | * |
| * | | |
| * * | | * |
+-------+ +-------+
You rolled a 7 on your first roll.
You win!
```
**Example #2.** After the title of the game is displayed (line 1), the player presses the <Enter> key to roll the two dice (line 3). Then, the visual representation for the dice roll of 2 and 1 is displayed on the screen (lines 3–7). Since a 3 was rolled on the first try, the player loses (lines 8–9).

```
Craps: A Popular Dice Game
Press <Enter> to roll the dice.
+-------+ +-------+
| *     | |     |
|       | |     |
|       | |     |
+-------+ +-------+
You rolled a 3 on your first roll.
You lose!
```

**Example #3.** After rolling the two dice (line 3), the visual representation for the dice roll of 4 and 2 is displayed on the screen (lines 3–7). Since 6 is neither 7 or 11 nor 2, 3, or 12, 6 becomes the point that the player must roll again to win before rolling a 7 (lines 9–12). The player rolls the dice again (lines 13–17). Since a 7 is rolled before the player rolls another 6, the player loses (lines 18–19).

```
Craps: A Popular Dice Game
Press <Enter> to roll the dice.
+-------+ +-------+
| *     | |     |
|       | |     |
| *     | |     |
+-------+ +-------+
You rolled a 6 on your first roll.
That's your point. Roll it again before you roll 7 and lose!
Press <Enter> to roll the dice.
+-------+ +-------+
| *     | |     |
|       | |     |
| *     | |     |
+-------+ +-------+
You rolled a 7.
You lose!
```
Example #4. 9 is established as the point. To win, the player must roll a 9 again without rolling a 7. The player wins by rolling a 4, 3, 3, and finally another 9.

Craps: A Popular Dice Game
Press <Enter> to roll the dice.

You rolled a 9 on your first roll.

That’s your point. Roll it again before you roll 7 and lose!

Press <Enter> to roll the dice.

You rolled a 4.

Press <Enter> to roll the dice.

You rolled a 3.

Press <Enter> to roll the dice.

You rolled a 3.

Press <Enter> to roll the dice.

You rolled a 9.

You win!
4. **Extra Credit: (8 points)** Write a Python program stored in a file `extra_credit.py` that analyzes and prints file statistics. First, prompt the user for the name of a file. Then, read and analyze the file and display the following statistics:

- total number of characters
- total number of sentences
- total number of words

Here are some notes regarding the above statistics you will collect.

- A character is any symbol in the text file. Examples of characters include letters, numerical digits, punctuation marks, and “invisible” characters such as whitespace and newline characters.
- Assume a word is separated by whitespace and/or newline characters.
- Assume sentences end in either a period, a question mark, or an exclamation point.

**Programming tips.** You will need to take advantage of the `strip()` and `split()` functions for strings.

**Example #1.** The user enters the filename `five-letter-words.txt` (line 1). The output of the program is then displayed (lines 3–5).

```python
Enter a filename: five-letter-words.txt
Characters: 16500
Sentences: 0
Words: 2750
```

**Example #2.** The user enters the filename `gettysburg.txt` (line 1). The output of the program is then displayed (lines 3–5).

```python
Enter a filename: gettysburg.txt
Characters: 1447
Sentences: 10
Words: 267
```
Submitting Your Assignment

Once you have completed your programs, submit each of them (q1.py, q2.txt, q3.py and extra_credit.py) electronically. The extra credit question is optional. You may resubmit your files as many times as you need until the due date. Only the most recent submission is graded.

You are required to include the following lines in the header of all your files:

```python
# File: filename.py
# Author: Student name
# Date: xx/xx/2019
# Section: Student section number
# E-mail: student_email@tamu.edu
# Description:
# e.g. This program asks for ...
```

- If you are in section 501 or 502, submit your files only on: [mimir.io](http://mimir.io)

- If you are in section 503, 504, 505 or 506, submit your files only on: [gradescope.com](http://gradescope.com)