HOMEWORK FOR LECTURE 10/12/2017
DUE 10/19/2018 [Again, we will only grade Q#3]

PRELIMINARIES:

Use <*gp9_exons.fasta*> for #2-3
Use <*watson_crick_paper.txt*> for #4

- Output format (unless noted otherwise) is flexible, but should be easily understood.
- Upload script for Q3 (and other Q's if you want) to Dropbox as ONE (1) zipped file.
- Make sure your name is part of the zipped filename

1. Use a while loop to repeatedly ask for a person's (one word) input
   - Count the results using a dictionary
   - Use no input as the signal for ending (i.e., just hitting return/enter)
   - OUTPUT: word and word counts, something like:
     - Word: number

##
## You can do problems 2 and 3 in one script if you wish ##
##

2. Giant files may cause memory problems.
   - Use the preferred way to read a giant file line-by-line to:

   - Make a better version of your exon reverse complementing file from the previous homework.
     - But this time using a fasta file: <gp9_exons.fasta>
     - (Do not need to concatenate into a cDNA)
     - This means you have to distinguish between the 'sequence name' and real sequence.
       - And possibly empty lines (i.e., line counting is NOT a good idea)
       - For example, you can see if '>' is in the line (indicating the name/ID of the sequence)
     - Output reverse complemented data into a fasta file
       - For example you could automatically name the file <gp9_exons.fasta.revcomp> by appending '.revcomp'

   - Use command line arguments to pass in the file name.
     - This way you script can do any fasta file

   - In addition:
     - Print out to the screen the total number of sequences in the file

3. Build on #2 and convert the reverse complementing part into a function
   - Functions usually go at top of script/program file. Main body follows below.

4. Write a script to count the occurrences of each word from a text file.

- Implement using a dictionary.
- Report the most and least (but at least 1x) often used word (or words if there are ties).
- The user enters the filename via the command line as first argument
- The user can input 0 (zero) or 1 word on the command line to have its count reported
  - If nothing, then output the default (most and least)
  - If user inputs 1 word, then first output default information,
    - Then data for word of interest.

- e.g., if you want to know “nucleic”

  -
    ```
    text_count_reporter.py watson_crick_paper.txt nucleic
    ```

- e.g., if you want to know “happy”

  -
    ```
    text_count_reporter.py feelgoodstory.txt happy
    ```

- Again output is flexible but should be “quickly” understandable to others, say your Boss.
  - e.g. Possible output (but you can be creative):

```
Examining file [filename.txt]
The most used word was: [the] used 101 times
There were 57 words used 1 time: [bollocks, grind, x, y, ..., z]
--------
You asked for the word count for:
happy: 5
--------
done
```