Assignment 4 :: 50 points :: Due: 11:59 p.m. Friday, 19 February

Objectives

This programming assignment will give you the opportunity to continue practicing with Python and to learn about the GenBank file format.

Introduction

The GenBank file format is the standard display format for the GenBank databases. The format consists of a set of nested keyword identifiers (written in all uppercase) and free-text entries. GenBank records end with a pair of forward slashes (`//`) on a single line. A sample record, with detailed descriptions of the fields, can be found here (*http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html*). Note that the only whitespace characters in GenBank records are spaces; there are no tabs.

An abbreviation of the record for NM_000492.3 is shown below. (Ellipses, `. . .`, represent portions of the record that have been omitted for brevity.) The complete record can be found here (*http://www.ncbi.nlm.nih.gov/nuccore/NM_000492.3*).

```
LOCUS       NM_000492               6132 bp    mRNA    linear   PRI 27-AUG-2019
DEFINITION  Homo sapiens cystic fibrosis transmembrane conductance regulator
            (ATP-binding cassette sub-family C, member 7) (CFTR), mRNA.
ACCESSION   NM_000492
VERSION     NM_000492.3
KEYWORDS    RefSeq; RefSeq Select.
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
            Catarrhini; Hominidae; Homo.
REFERENCE   1  (bases 1 to 6132)
. . .
ORIGIN
        1 aattggaagc aaatgacatc acagcaggtc agagaaaaag ggttgagcgg caggcaccca
       61 gagtagtagg tctttggcat taggagcttg agcccagacg gccctagcag ggaccccagc
      121 gcccgagaga ccatgcagag gtcgcctctg gaaaaggcca gcgttgtctc caaacttttt
      181 ttcagctgga ccagaccaat tttgaggaaa ggatacagac agcgcctgga attgtcagac
      241 atataccaaa tcccttctgt tgattctgct gacaatctat ctgaaaaatt ggaaagagaa
      301 tgggatagag agctggcttc aaagaaaaat cctaaactca ttaatgccct tcggcgatgt
      361 tttttctgga gatttatgtt ctatggaatc tttttatatt taggggaagt caccaaagca
      421 gtacagcctc tcttactggg aagaatcata gcttcctatg acccggataa caaggaggaa
      481 cgctctatcg cgatttatct aggcataggc ttatgccttc tctttattgt gaggacactg
      541 ctcctacacc cagccatttt tggccttcat cacattggaa tgcagatgag aatagctatg
      601 tttagtttga tttataagaa gactttaaag ctgtcaagcc gtgttctaga taaaataagt
      661 attggacaac ttgttagtct cctttccaac aacctgaaca aatttgatga aggacttgca
. . .
     6001 ccaccagtct gactgtttcc atcaagggta cactgccttc tcaactccaa actgactctt
     6061 aagaagactg cattatattt attactgtaa gaaaatatca cttgtcaata aaatccatac
     6121 atttgtgtga aa
//
```

It is important to note that the sequence portion of the record follows the keyword **ORIGIN** and that **ORIGIN** is always the last keyword in the record. Note also that the numbers on the sequence lines are position numbers and are not part of the actual sequence.

Program Description

Write a (procedural) Python (version 3) program, **netID_Assign4.py** where **netID** is your UNO NetID, which converts a multi-record GenBank file into a multisequence FASTA-formatted file. Records in the output must come one right after another—they should not be separated by a blank line (or lines).

The format of the FASTA header should be:

>accession|accession-number| locus

for Genbank records. Note that there is a single space following the last pipe (`|`). Sequence should be converted to uppercase and split into multiple lines of width specified by a command-line option (see below). This conversion should be done in a function called **formatSeq()** that takes two arguments, the sequence and the width, and returns the formatted sequence.

As an example, the above record converted to FASTA format (60-character sequence width) would look like the following:

```
>accession|NM_000492.3| Homo sapiens cystic fibrosis transmembrane conductance regulator (ATP-
binding cassette sub-family C, member 7) (CFTR), mRNA.
AATTGGAAGCAAATGACATCACAGCAGGTCAGAGAAAAAGGGTTGAGCGGCAGGCACCC
AGAGTAGTAGGTCTTTGGCATTAGGAGCTTGAGCCCAGACGGCCCTAGCAGGGACCCCA
GCGCCCGAGAGACCATGCAGAGGTCGCCTCTGGAAAAGGCCAGCGTTGTCTCCAAACTT
TTTTTCAGCTGGACCAGACCAATTTTGAGGAAAGGATACAGACAGCGCCTGGAATTGTC
AGACATATACCAAATCCCTTCTGTTGATTCTGCTGACAATCTATCTGAAAAATTGGAAA
GAGAATGGGATAGAGAGCTGGCTTCAAAGAAAAATCCTAAACTCATTAATGCCCTTCGG
CGATGTTTTTTCTGGAGATTTATGTTCTATGGAATCTTTTTATATTTAGGGGAAGTCAC
CAAAGCAGTACAGCCTCTCTTACTGGGAAGAATCATAGCTTCCTATGACCCGGATAACA
AGGAGGAACGCTCTATCGCGATTTATCTAGGCATAGGCTTATGCCTTCTCTTTATTGTG
AGGACACTGCTCCTACACCCAGCCATTTTTGGCCTTCATCACATTGGAATGCAGATGAG
AATAGCTATGTTTAGTTTGATTTATAAGAAGACTTTAAAGCTGTCAAGCCGTGTTCTAG
ATAAAATAAGTATTGGACAACTTGTTAGTCTCCTTTCCAACAACCTGAACAAATTTGAT
GAAGGACTTGCA. . .CCACCAGTCTGACTGTTTCCATCAAGGGTACACTGCCTTCTC
AACTCCAAACTGACTCTTAAGAAGACTGCATTATATTTATTACTGTAAGAAAATATCAC
TTGTCAATAAAATCCATACATTTGTGTGAAA
```

where the ellipsis (. . .) represents a portion of the sequence that has been omitted for brevity.

So that your program works with files of arbitrary size, do not read in the entire file at once (i.e., using `.readlines()`). Instead read in the file one line at a time (i.e., using `.readline()`).

The program should take three command-line options as follows:

Option	Function
i	Specifies the name of the input file. If this option is not used, input should be taken from standard input (i.e., the keyboard).
o	Specifies the name of the output file. If this option is not used, output should be sent to standard output (i.e., the screen).
l ("el")	Specifies the width to which the sequence should be formatted. If this option is not used, the sequence should default to 70 characters in width.

Using a function (`usage()`), your program should display usage information to the screen if an option at the command-line is not recognized.

Hint: Write your program in stages making sure that each stage works before you start work on the next. For example, first write the program to process a single-record file, then modify it to accept command-line arguments and finally modify it to process a multisequence file.

Testing Your Program

Example files `NM_000492.gb` (contains a single GenBank record), `sequences.gb` (contains multiple GenBank records) and `sequences.fna` (an example output file; 70-character sequence width) can be copied from Canvas under `Files/Assignments/Assignment04`.

The Unix `diff` command compares two files line by line and displays those lines that are different. You can use this command to test the output of your program to make sure that it is working correctly. For example, if your output file is called `output.fna` you could compare it with mine (`sequences.fasta`) as follows:

```
[bioimav@bioi3500]$ diff sequences.fna output.fna
[bioimav@bioi3500]$
```

As illustrated in this example, if there are no differences between the two files no output will be displayed and you will immediately get the Unix prompt back. This is one of the ways I will check your program. You will only get full credit if there are no differences between your output file and mine.