These questions are intended to extend your understanding of pointers and to develop an understanding of structs and hash tables. You will also practice use of binary files and working with big data. (We will discuss installation and use of CMPH.)

1. (`C` code) In this question, you will read in a large dataset over 1Gb large into memory. You will create a hash table to facilitate operations on the data using the CMPH Library. Finally, you will rewrite the data as a binary file to facilitate further work with these data. Make sure you have the CMPH library installed on your container before you start Part (b).
   (a) The data are in fastq format, a common format of very large data files produced by modern sequencing machines (a future version of which may some day "sequence your genome" for a fee). Each set of four lines in the fastq correspond to a single read from the sequencing machine. The first line provides a name for the read, the next line the nucleotide sequence of the read, and after a separator line, the last line the quality scores of the read, encoded in ASCII format. Design a structure to store the reads and read the entire contents of the data file provided via Canvas into memory using an array of these structures.
   (b) Create a hash table for the reads by hashing on the nucleotide sequence, *i.e.* the nucleotide sequence is the key in the associative array. The value in the hash table should store:
   - the number of reads with the same unique nucleotide sequence, and
   - the indices of these reads in the array from Part (a).

   (Hint: It may be beneficial to only count the sequences while creating the hash. A second pass through the data associates the appropriate read indices with each key in the hash.)
   (c) It is likely that processing the file and creating the hash takes some time, so to facilitate further rapid analysis of these data, choose a binary format for the fastq data (not the hash). Put what you need in the binary format, where you need it. It is your choice, and you should make your life easier. Write a `read_bfastq()` function to read and a `write_bfastq()` to write your binary fastq format. Write the data to file. Also write the hash to file using the cmph library io functions.

2. In this question, you will focus on the counts of each unique sequence. This information should be in your hash table, which is now stored in a file that you can load with cmph library io functions.
   (a) Let X be the number of times a unique sequence is observed in the fastq file. One possible distribution for these count data is
   $$p(x) = \begin{cases} p_x & x < x_{\min} \\ \frac{x^{-\alpha}}{\zeta(\alpha, x_{\min})} & x \geq x_{\min}, \end{cases}$$
   where p_x are probabilities such that $\sum_{x<x_{\min}} p_x + \zeta(\alpha, x_{\min}) = 1$, and
   $$\zeta(\alpha, x_{\min}) = \sum_{n=0}^{\infty} \frac{1}{(n + x_{\min})^{\alpha}} = \frac{1}{\Gamma(\alpha)} \int_0^{\infty} \frac{t^{\alpha-1} e^{-x_{\min} t}}{1 - e^{-t}} dt. \tag{1}$$
   Given $x_{\min}$, find an equation for which the MLE $\hat{\alpha}$ is the root.
   (b) (`C` code) The integral in (1) should behave well (most of the time) and converge faster than the series, by replacing the infinite upper limit with some large M. (If it doesn't for these data, write the code and diagnose the problem. We'll build the tools that can deal with this problem later.) Given the count data $x_1, x_2, \ldots, x_n$ for the unique sequences in the fastq file, use your bisection algorithm (or something better if we get to it) to find the MLE $\hat{\alpha}$ given $x_{\min} = 1$.
   (c) (`C` code) Now maximize the Bayesian Information Criterion
   $$\text{BIC} = 2 \sum_{i=1}^{n} \ln p(x_i) - x_{\min} \ln n$$
   to choose $x_{\min}$. (For each choice of $x_{\min}$, you should maximize α as per Part (a) of this question.)

(d) (`C` code) The quality scores encode the probability of error in each nucleotide of the read (recall the Poisson-Binomial homework?). The quality score is converted to a probability of error as

```
char q = ',';        /* example */
double p = pow(10, (q - 33) / 10.);
```

Assuming the errors are independent across sites in the read, compute the expected number of *error-free* reads of each unique sequence, which is something less than the observed count since some of these contain errors. With these *continuous* data $y_1, y_2, \ldots, y_n$, one for each unique sequence, compute the MLE

$$\hat{\alpha} = 1 + n \left(\sum_{i=1}^{n} \ln \frac{y_i}{y_{\min}} \right)^{-1}$$

for the continuous version of $p(y)$ with $y \geq y_{\min}$. Set $y_{\min} = 1$.