Test Engineer Challenge

Welcome to the DNAnexus test engineer challenge! This challenge is focused on your skills as a test engineer, as well as your general programming ability.

If you find anything confusing, just try to solve the problem as best you can. There isn't necessarily a single "right" answer - we are interested in seeing how you approach these types of problems, as well as your ability to write clean, correct code.

You may write the code in the language of your choice (please, nothing too exotic!), but it should be running code, not pseudocode. You're free to use standard library functions of the language you're using, but don't use a function that directly solves the problem (use your judgement). Please send us your original source code, preferably with comments explaining what it's doing. Although it's not necessary, if you also created any test stubs and test data, feel free to send that as well.

If anything about the following questions is unclear, or if you have any feedback, please let us know! Thank you for your time and interest in DNAnexus, and good luck!

Problem: DNA sequence conversion

DNA is a long molecule that lies inside the nucleus of a cell, and it can be thought of as a very long string consisting of characters in the alphabet {A, C, G, T}. DNA sequencing is the technology that enables reading from DNA molecules and converting them to strings on the output. We are interested in a technology that works in the following way: the DNA molecules in the input are fragmented into pieces of equal length L; each piece is then sequenced by the technology, and its content is encoded in the output. The particular encoding used in the output is the following:

- The file contains multiple consecutive entries, one per piece.
- Each piece is represented by L consecutive bytes (1 byte = 8 bits).
- The first two (most significant) bits of each byte encode the DNA letter:
    - 00 represents A
    - 01 represents C
    - 10 represents G
    - 11 represents T
- The last six (least significant) bits of each byte encode the confidence that the readout was correct, also known as the quality score. It is represented as an unsigned 6-bit integer in the range 0 to 63.

Write a program whose inputs are:

- The path to an encoded file
- The number L

An example input file (named "**input**") should've been provided to you along with this document. The example file has 1680 bytes.

Before you begin, if you wish to check the MD5 message digest of the input file, the expected value is shown below:

MD5 (input) = 25a02f1331042be2856e652bda60e8de

One can generate different output with different L values. Example output files for L = 7, 15, and 80 should've also been provided to you along with the sample input file.

The program should print on standard output the contents of the input file, presented in the following format (known as the FASTQ format):

- Each piece is represented by four lines:
  - The first line contains the word **@READ_** followed by the piece index. The first piece has an index of 1, so its first line would be **@READ_1**
  - The second line contains L characters in the {A,C,G,T} alphabet, representing the DNA sequence of the piece.
  - The third line contains the word **+READ_** followed by the piece index (e.g., **+READ_1**).
  - The fourth line contains L characters, representing the quality scores of the piece. Each score is represented as an ASCII character in the range 33-96, by adding 33 to the original score. For example, if the original score is 0, it should be represented by the ASCII character 33 ("!")

For example, if the binary input file contains this data:
00000000 11100000 11000001 01111111

The output (for L=2) would be:
@READ_1
AT
+READ_1
!A
@READ_2
TC
+READ2
!`