(10 points) Problem 1 Download the "amazonReviews.json" file from Canvas. Each data point in the data set has the following syntax:

"reviewerID":,"asin":,....,"reviewText":,"overall":,....,"reviewTime":

1. Load the JSON file into your choice of labeled data structure.
   (A JSON file is a file that stores simple data structures and objects in JavaScript Object Notation (JSON) format, which is a standard data interchange format)

2. Discard all the columns except ('reviewerID', 'reviewText') since we won't be using them in this assignment.

3. Process the reviews by removing any punctuation marks, stop words (stopwords list), and converting the text to lower case.
   (You can use any data structure that you think will be the most appropriate for accomplishing the above tasks.)

(10 points) Problem 2. To compare the reviews, you should represent the reviews appropriately.
Convert each review into a set of k-shingles. Remember, the set of k-shingles for any string is simply the set of all sub-strings of length k in the string. You will likely have to experiment a bit to find a good value of k. Also think of how to handle spaces in the text.
Represent the reviews in terms of binary matrix.

$$\begin{bmatrix} 0 & 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ & \vdots & & & \ddots & \\ 1 & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}$$

Hint: Good values for k tend to be between 4-10. You may also have to either add padding to or remove reviews that are too short for your shingle length.

(10 points) Problem 3. To get a sense for the data, let's analyze a random sample. Pick 10,000 pairs of reviews at random and compute the average Jaccard distance and the lowest distance among all pairs. Plot a histogram of the pairwise Jaccard distances, and include all results in your report.

(20 points) Problem 4. Our final goal will be to find the approximate nearest neighbor of a queried review. However, the current shingle representation is not the most efficient way to store the data for this purpose. Find a more efficient way to store the data that you may use for the later problems.

Hint: Think of the data structures introduced in class and the implementation should be as efficient as possible

(30 points) Problem 5. Using your data structure from the previous problem, detect all pairs of reviews that are close to one another.
We define two reviews to be *close* if their Jaccard distance is below 0.2. You should do this as efficiently as possible, both in terms of storage and time complexity, using the entire data set. If your solution requires you to pick any parameters, you should justify your choice with plots or data in your report, whichever is appropriate. Dump all such pairs to a CSV file and include it in the same zip folder.

(15 points) Problem 6. Create a function that accepts a queried review and returns its approximate nearest neighbor (reviewerID). This should be done as efficiently as possible.

(5 points) Problem 7. Briefly discuss the complexity of your implementation and how it is better than the naive implementations.