1. This question is dry (does not require programming). In each section, you are required to describe the solution in terms of technologies learned during the course, including the schema, the data flow sketch and the pseudo code. Optimize your solutions wherever possible – the goal is minimizing the latency of receiving the result.

Consider a large website with multiple content categories (news, sports, entertainment, etc). Each page belongs to one category. The site’s operation team is interested to track the user traffic statistics. For this purpose, it maintains two datasets: (1) the `pages` table, which holds for each page its `url` and `category`, and (2) the `views` table, which holds for each page view event (download) the user id (`cookie`), the page `url` and the `access time`. You can assume that the `views` size exceeds the `pages` size by orders of magnitude.

   a. Compute, in two Map-Reduce jobs, the aggregate (overall) number of page views by category over the last 7 days.

   b. Assume that the page dataset is fixed, and the data about the new views is received once a day. Describe an incremental process to compute the number of views per page over the last 7 days.

   c. Compute, every night as the new views dataset arrives, the average and variance of the number of views per day per page from the dawn of history.

2. This question is a follow-up to Q3 in HW 1. A company implemented your solution to that question for computing the Jaccard similarity coefficient between every two users, and ran it nightly over the entire input of \(\langle\text{user-id, movie-id}\rangle\) tuples it had accumulated.

   After a year, the computation started taking quite long. The task of this question is to transform the computation from being a long batch process over the entire input, to being a lighter incremental process. Concretely, given all historical input, the similarity coefficients computed at midnight of date T, and the \(\langle\text{user-id, movie-id}\rangle\) tuples of day T+1, compute as efficiently as possible (in pseudo-code) the Jaccard similarities at midnight of date T+1. Explain your design choices, and detail the complexity of your nightly process.

   In addition to the assumptions given in Q3 of HW 1, you may assume the following:
   
   - Users never see the same movie twice.
   - For any movie m, the number of users who view m on day T+1 is less than an \(\varepsilon\)-fraction of the number of users who viewed m during days 1,\ldots,T.

3. Compare the COGROUP operator in Pig (see the original paper) with the join operator in SQL presented in the class. Explain the rationale for the difference. Provide your own example in which the use of COGROUP is beneficial.
4. This question relates to slides 18-22 of Lecture 7.
   a. Slide 21: prove that the greedy visit strategy indeed visits every node in \( G' \) infinitely often.
   b. Prove that the “anti greedy” visit strategy, that visits the page with the least amount of positive (i.e. greater than zero) cash, does not meet the fairness condition specified on slide 18.
   c. Slides 20, 22: modify the Theorem to prove the convergence of \[ \frac{[B(v)+C(v)]}{[G+1]} \] to \( \pi \).

5. This question relates Bloom filters presented in Lecture 8.
   Let \( U \) be a universe of elements, and let \( B \) be a Bloom filter of size \( m \) bits that can accommodate \( n \) element insertions while having false positive probability \( \varepsilon \). We would like to add support of deletions to this Bloom filter. Using only \( m \) extra bits, devise a solution that will add a false negative probability of \( \varepsilon \) (and will even slightly decrease the false positive probability). Prove your answer. You may assume that deleted elements are not reinserted.

**Good Luck!**