Hint for Q3 (In-memory Join)

Please see some notes below that may be helpful for solving Question 3:

In this question, you will need both files soc-LiveJournal1Adj.txt and userdata.txt since the states and names of all users are in userdata.txt.

To perform this task, you can implement the simplest version of in-memory join, since the userdata.txt is small enough to completely fit in memory on each node.

So, the idea is to load userdata.txt dataset into memory in every mapper, using a hash map data structure to facilitate random access to tuples based on the join key (userid). For this purpose, you can override the method setup (mapper initialization) inside the Map class and load the hash map there inside.

Note that you can pass the path of your userdata.txt as an argument for your program or just hardcode the path inside your java code (make clear how you coded it in your README file on you submission).

Mappers are then applied to the larger input (soc-LiveJournal1Adj.txt), and for each one of the userid of a friend, pair the mapper probes the in-memory dataset (userdata.txt) to see if there is a tuple with the same join key.

You are allowed to use job chaining technique (from output of Q1), but a better approach would be to change a little bit your solution for Q1 without necessarily using job chaining.

Just one last note: please observe that the output provided as sample in the assignment is just for illustration purpose

26, 28 [Evangeline: Ohio, Charlotte: California]

Evangeline and Charlotte and not necessarily common friends of userids 26 and 28.