## *Algorithm/Pseudocode*

1. Read in the input data
2. Split data on “\t” character—this separates the UserID from the list of friends
3. Split the list of friends by “,” character, filter out empty strings, and convert to ints
4. Create a dataframe based on this new filtered data
5. Change the dataframe so that we get pairwise relationships instead of an array of friends
6. Join the dataframe with itself so that we can start finding friend recommendations
   1. Select columns that represent the user, the friend recommendation, and the friend that they share in common
   2. Filter out self-recommendations (i.e., the friend recommendation should *not* be the same as the user)
7. Count the friend recommendations
8. Get rid of existing friends so that we don’t recommend them again
9. Get the top 10 recommendations for each user
10. Format the output to display in the next step
11. Randomly sample 10 users and display their friend recommendations

## *Explanation*

Steps 1-4 are self-explanatory. Here we read in the input data and split it based on its format. The format of the input file is formatted in the following way: the user’s ID, a tab character, and a comma-separated list of other IDs that represent that user’s friends. Based on the assignment’s instructions, we can implement RDDs or dataframes; I chose the latter.

In step 5, we change the dataframe to have pairwise relationships between users and their friends. To give a more visual representation of what we’re doing:

user | friend instead of user | friend

0 1 0 [1, 2,…]

0 2

These pairwise relationships will make it easier to process later on.

Step 6 joins this newly formed dataframe with itself. We’re doing this because we want to find friend recommendations based on user’s mutual friends. We also select the columns we want as the output of a new dataframe. The new dataframe will have the columns “user,” “friend recommendation,” and “friend in common.” As the names suggest, “user” represents a specific user, “friend recommendation” represents a friend that was recommended to them, and “friend in common” represents the friend that causes the two users to be recommended to each other. Lastly, we filter out self-recommendations, meaning that we don’t want users to be recommended to themselves.

Step 7 counts the friend recommendations between each pair of users. This’ll be used to get their top 10 recommendations.

In step 8, we get rid of existing friends that the user already has because we want to avoid recommending them again.

Finally, steps 9-11 get the top 10 recommendations for the user, based on the number of friend recommendations they have; format the output in a specific way (based on the assignment instructions); and take a random sample of 10 users from the input file and outputs their top 10 friend recommendations.