1. Select a 'starting point,' i.e. a user on Twitter, which could be yourself or somebody else.

2. Retrieve his/her friends, which should be a list of id's, and followers, which is another list of id's, perhaps using the **get_friends_followers_id()** function from the Cookbook, or your own program if you prefer. Note: When you use get_friends_followers_id() or its equivalent, you are allowed to set the maximum number of friends and followers to be 5000 (but no less), in order to save API calls, and hence your time.

3. Use those 2 lists from Step 2 to find **reciprocal friends**, which is yet another list of id's. (The definition of 'reciprocal friends' can be found in my slides.) These are the **distance-1 friends**.

4. From that list of reciprocal friends, select **5** most popular friends, as determined by their **followers_count** in their user profile. (I suggest you use the **get_user_profile()** function from the Cookbook to retrieve the user profiles of the reciprocal friends.)
5. Repeat this process (Steps 2, 3 & 4) for each of the distance-1 friends, then distance-2 friends, so on and so forth, using a **crawler**, until you have gather at least **100** users/nodes for your social network. Note: I suggest you modify the crawler (**crawl_followers()**) function from the Cookbook or my simplied crawler to do this. However, please note that either one of these 2 crawlers retrieves only followers. You need to modify it to get both followers and friends, in order to compute the reciprocal friends .

6. Create a social network based on the results (nodes and edges) from Step 5, using the **Networkx** package, adding all the nodes and edges.

7. Calculate the **diameter** and **average distance** of your network, using certain built-in functions provided by Networkx (in 3.22 Distance Measures & 3.45 Shortest Paths, or your own functions if you prefer.