1.

```
File Name: twitter.txt(KEY)
File Content: value
map(String key, String value):
    //declare lists to store user's and their counts
    declare list[]

    //loop through each line of the key file(twitter.txt)
    for each pair p in value:
        //split user & follower
        split(p) => [follower, user]
        //insert user into list and set count to 1.
            //check if user already exists in list
            if user exists in list then
                //increment count for that user
                increment list['user']
            else
                //set count to 1
                list['user'] = 1
    // After generating list pass it to reduce function

--------------------------------------------------------------------------------
value: set of user's and their follower counts.
reduce(list value):
    //declare one output object json/list.
    declare output[]

    //loop through each user:count pair
    for each element ele in value:
        //split user & count
        split(ele) => [user, count]
        create string as `count, user`
        insert to output list
    print output
```

2.

As we know that Distributed Programming Systems are deployed in clouds to handle large data set problems. Even with Distributed Programming systems there are challenges to be faced like Failure in servers, Synchronization issues, Ordering & state management etc.,

In order to resolve these challenges MAP Reduce programming system came into picture. It processes and generates large datasets and resolves intermittent issues on its own.

Map Reduce can handle thousands of processes at a time as well as parallelization. It also takes care of I/O scheduling, status monitoring and Fault tolerance.

In the case of a single computer program that is running in multiple virtual instances doesn't solve the large data set problems. Since it's a parallel processing, we can face issues similar to the above mentioned.

The time complexity of program running in vm's is more compared to the Map reducing in the case of huge amounts of data.

3.

```
// --> Start
/**
 * need to find the { 'user': [follower List] }
 */
File Name: twitter.txt(KEY)
File Content: value
map(String key, String value):
    //declare lists to store user's and their counts
    declare list[]
    for each pair p in value:
        //split user & follower
        res-> split(p) => [follower, user]
        // check if already key exists then push into the followers array
or else create new array with key as user.
```

```
        insert res[0] into list[res[1]]
    //Final list should be like this below.
    /**
     * list = [ "user1":[ 'follower1', follower2, ...],
     *          "user2":[ 'follower1', 'follower2', ....], ]
     *  */
    call reduce(list)
    //end of map function
//--------Reduce Function starts from here-----------------------
/**
 * Loop through each user and their followers and then find the common
follower list
 */
reduce(list value):
    declare outputList[]
    //loop through users
    for each user1 in value:
        // start loop from next user to end
        for each user2 > index(user1) in value:
            count = 0
            // start looping through the followers and check for common
and count them.
            for follower in user2:
                if follower exists in user:
                    increment count++
            key = `user1, user2`
            outputList[key] = count

    print(outputList)
/**
 * Final outputList contains the list like this below
 * outputlist = [
 *          'user1, user2': `commonFollowerCount`,
 *          'user1, user3': `commonFollowerCount`,
 *          .
 *          .
 *          .
 * ]
 */
```