CMSC132 Fall 2018 Maps Worksheet

1. Rewrite the WordFrequencyCounter.java lecture example so the case of words is ignored. To rewrite it use the TreeMap constructor that takes a Comparator as a parameter. You don't need to write a Comparator class (use the String.CASE_INSENSTIVE_ORDER comparator).

2. Discuss how you would use a hash table to implement a map.

3. Define a class called Friends based on specifications below. The **Friends** class will keep track of a person's friends.

    a. Instance variables
        i. Map<String, Map<String, Boolean>> friends → This map will be used to store information about the friends a person has. The key represent a person and the value is a map that indicates whether a person is a friend of another person.
        ii. Any other instance variables you think you need.

    b. Methods (loosely defined, which means feel free to add / modify parameters)

        i. **Constructor** – Creates the appropriate map and performs any other required initialization.

        ii. **defineFriendship(String person, String toPerson, boolean isFriend)** → This method will define or update the relationship that exist between **person** and **toPerson**. For example, defineFriendship("Kelly", "John", true) defines "John" as a friend of "Kelly"; defineFriendship("Kelly", "Kyle", false) defines "Kyle" as not a friend of "Kelly". The method will create a map for the specified **person** if one does not exist; otherwise it will use the current map. Keep in mind that if person is a friend of toPerson, toPerson is also a friend of person (two entries must be added/updated in the map).

        iii. **boolean areFriends(String person1, String person2)** – Returns true if two people are friends and false otherwise.

        iv. **toString()** – For each person, it returns the name of the person followed by their friends. The information must be sorted alphabetically. Here is a sample string:

        Abby (Al, Bob)
        Al (Abby, Bob)
        Bob (Abby,Al)
        Tom ()

        v. **boolean couldBeFriends(String person1, person2)** – Returns true if **person1** could become a friend of **person2**. To determine whether a person can become a friend of another, you need to determine whether the is a chain of friends starting at person1, leading to Person2. For example, if "Peter" is a friend of "Jenny", and "Jenny" is a friend of "Tom", then "Peter" could be a friend "Tom".