Problem 1 - Twin Words

Two words are defined as "Twin Words" of each other if they satisfy:-

1. Same number of consonants (letters which are not vowels like b,c x etc.) in both words.
2. Same set of consonants in both words

A phrase is said to be a “Twin Phrase” if all the words of a phrase are “Twin Words” of any word in the other phrase

For example

1. BBC and BCC are Twin words as they have the same set of consonants - b and c; and the same number of consonants - 2
2. Red and bread are not Twin words

Cover the following cases -

1. Write a program to check if two words are "Twin words" of each other
2. Write a program to take an input file of list of words (one word per line) and print out sets of "Twin Words"
3. Now assume that input to previous question is a list of phrases instead of words. And a phrase is said to be a "Twin Phrase" if all the words of a phrase are a "Twin Word" of any word in another phrase. Write a program to print out sets of "Twin Phrases"
   For eg. “Red BBC” and “Dear rad BCC” are twin phrases as every word in phrase 1 has a twin word in phrase 2 and vice versa

Submit your solution in one single file covering all 3 cases. Upload your code in a language of your choice and name the file "peak_java", "peak_python" etc. If you couldn't code, explain the logic and upload your logic in a text file. We want to read through your solution.

Problem 2 - Peak Interaction

The log file contains interaction between users on Toppr in a specific format. We want to find a group of users communicating among each other.

A group is a set of at least three users, where every possible permutation of two users within the group have both received and sent some kind of interaction between the two.

Input specifications:

The input file consists of multiple lines of aggregated log data. Each line starts with a date entry, whose constituent parts are separated by single white spaces. The exact format of the date always follows the examples given below.

Following the date is a single tab, and then the email address of the user who is performing the action. Following that email is another single tab and then finally the email of the Toppr user who receives the action. The last line of the file may or may not have a newline at its end.

Example input file:

```
Thu Oct 18 16:53:01 PST 2019    a@toppr.com    b@toppr.com
Thu Oct 18 16:53:02 PST 2019    b@toppr.com    a@toppr.com
Thu Oct 18 16:53:03 PST 2019    a@toppr.com    c@toppr.com
Thu Oct 18 16:53:04 PST 2019    c@toppr.com    a@toppr.com
Thu Oct 18 16:53:05 PST 2019    b@toppr.com    c@toppr.com
Thu Oct 18 16:53:06 PST 2019    c@toppr.com    b@toppr.com
Thu Oct 18 16:53:07 PST 2019    d@toppr.com    e@toppr.com
Thu Oct 18 16:53:08 PST 2019    e@toppr.com    d@toppr.com
Thu Oct 18 16:53:09 PST 2019    d@toppr.com    f@toppr.com
Thu Oct 18 16:53:10 PST 2019    f@toppr.com    d@toppr.com
Thu Oct 18 16:53:11 PST 2019    e@toppr.com    f@toppr.com
Thu Oct 18 16:53:12 PST 2019    f@toppr.com    e@toppr.com
```

Every line in the input file will follow this format, you are guaranteed that your submission will run against well formed input files.

Output specifications:

You must output all groups detected from the input log file with size of at least 3 members. A group is defined as N >= 3 users on Toppr that have send and received actions between all possible permutations of any two members within the group.

Your program should print to standard out, exactly one group per line. Each group must have its member user emails in alphabetical order, separated by a comma and a single space character each. There must not be a comma (or white space) after the final email in the group; instead print a single new line character at the end of every line. The groups themselves must be printed to standard out also in alphabetical order; treat each group as a whole string for purposes of alphabetical comparisons. Do not sort the groups by size or any other criteria.

Example output (newline at end of line):

```
a@toppr.com, b@toppr.com, c@toppr.com
d@toppr.com, e@toppr.com, f@toppr.com
```

Finally, any group that is a sub-group (in other words, all users within one group are also present in another) must be removed from the output. For this case, your program should only print the largest super-group that includes the other groups. Your program must be fast, efficient, and able to handle extremely large input files.

Upload your logic or code in a language of your choice. Name the file "peak_java", "peak_python" etc. If you couldn't code, explain the logic and upload your logic in a text file. If your approach is close, we can take care of the rest!