Welcome to the Gracenote Music Engineering interview! Please complete all questions in Section I, and optionally, implement the programming challenge in Section II. If time allows, please send your solutions ahead of your in-person interview (to the person you received this email from), however your interview is coming up shortly then sending ahead is not required. Do try to solve at least Section I by the time of the in-person interview though, and please bring a printout of your source code to the interview.

I. PROGRAMMING QUESTIONS

In our job as Music Engineers, we live and breathe music metadata & information, as well as algorithms and APIs for processing, matching, cleaning, filtering, grouping, recommending etc. music data. The below questions will give us a sense of your technical prowess, while at the same time give you a tiny glimpse of the kinds of problems we're faced with on a daily basis.

Please write a program that solves the below questions. Extra points if you do this in more than one languages, and particularly if the languages include one compiled and one scripting language.

(Note: Python and C are the ones the Music Engineering team works in the most, but others are welcome too.)

The attached file "artistnames.txt" contains a few thousand randomly selected artist names. For simplicity, we have restricted to names in Latin scripts, and have converted the text to the simple ASCII character set by removing diacritics etc.

Write a program to:

1) Load the artist names from the file into memory.

2) Print out the list sorted by the length of the artist names, longest names first.

Example output lines:

```
86      Montserrat Caballe, Alfredo Kraus, Etc.; Jonel
Perlea: RCA Italiana Orchestra & Chorus
78      Ghostface Killah Feat. Raekwon, Sun God, Trife Da
God, Method Man & Cappadonna
50      Elvis Costello & The Attractions Feat. Jimmy Cliff
```

```
50      Jerry Lee Lewis Feat. Merle Haggard & James Burton
42      The University Of California Marching Band
41      Alfred Newman: 20th Century Fox Orchestra
16      Barenaked Ladies
16      Big Joe Williams
7       CASCADE
3       Air
```

3) Print out every unique word in the total list of artist names, along with the frequency count, ordered by frequency (descending). The logic should be case insensitive.

Example output:

```
500     feat
259     the
106     orchestra
47      of
43      dj
33      chorus
32      symphony
31      etc
27      philharmonic
27      david
27      john
25      la
23      b
23      de
22      band
21      paul
19      i
19      a
18      chris
17      t
17      vs
17      d
17      big
16      london
16      opera
16      j
```

4) Print out the artist names where the letters can be scrambled to create a palindrome, using all characters in the name (including spaces and punctuation). The palindrome does not have to mean anything or be pronounceable and the logic should be case insensitive. Extra points if you also print out one of the possible palindromes.

Example output:

```
311 => 131
995 => 959
Allez Allez => allez zella
Bazbaz => bazzab
CSS => scs
Ee => ee
H => h
ISIS => issi
JESSE => esjse
MAA => ama
Nene => neen
Otto => otto
Penny Penny => penny ynnep
Pollo => olplo
Pony Pony Run Run => ponny ru ur ynnop
Sing-Sing => sing-gnis
So..So => so..os
Temper Temper => teempr rpmeet
Wakey!Wakey! => wakey!!yekaw
wingenfelder:Wingenfelder => winngeeefldr:rdlfeeegnniw
```

5) Write an algorithm that recognizes and guesses artist collaborations, and prints out the guessed components.

Example output:

```
Jay-Z Feat. Beyoncé => Jay-Z (AND) Beyoncé
Funkstar De Luxe vs. Bob Marley => Funkstar De Luxe (AND)
Bob Marley
Indigo Girls, Jewel & Sarah McLachlan => Indigo Girls (AND)
Jewel (AND) Sarah McLachlan
```

Note: this is a hard problem to get right with a simple algorithm and with just having access to this data set, so we’re not looking for a perfect solution. We’re just looking for you to come up with some ideas of your own and to demonstrate them in code. Describe some more advanced ideas you can think of to get better at this problem.

II. PROGRAMMING CHALLENGE!

Note: this part is optional. If you choose to complete it, the language and environment is up to you.

"Tab Flip"

Tablature is a popular form of musical notation used by guitarists, which shows what number fret (0-24) to press on each string. Here is an example of what it looks like:

```
e|---0---1---3---
B|---0---1---0---
G|---1---2---0---
D|---2---3---0---
A|---2---3---2---
E|---0---1---3---
```

In this example above, there are 3 chords that are played in this order:

1st Chord: The E string is open (no fret is pressed), 2nd fret pressed on the A string, 2nd fret pressed on the D string, 1st fret pressed on the G string, B string is open, e string is open.

2nd Chord: 1st fret pressed on E string, 3rd fret pressed on A string, 3rd fret pressed on D string, 2nd fret pressed on G string, 1st fret pressed on B string, 1st fret pressed on e string

3rd Chord: 3rd fret pressed on E string, 2nd fret pressed on A string, D string open, G string open, B string open, 3rd fret pressed on e string

For this challenge, write a program that reads in a simplified guitar tab file consisting of 6 lines and outputs the frets that are pressed on each string in a comma separated file. See input and output examples below. We should be able to run your program and specify the path to the tab file and the output file as command line parameters.

Examples:

```
python tab_flip.py /path/to/tab/file /path/to/output
./tab_flip /path/to/tab/file > path/to/output
```

NOTE: the input tablature file will always contain six lines and will contain the string names followed by "|" character as shown in the example. Beyond that, the tablature file will only contain “-“ characters and numerical values in the range of 0 to 24 representing the frets. Also, there will always be at least one “-“ character separating fret values.

```
Example Input:
e|---0---1---3---
B|---0---1---0---
G|---1---2---0---
D|---2---3---0---
A|---2---3---2---
E|---0---1---3---
```

```
Example Output:
E,A,D,G,B,e
0,2,2,1,0,0
1,3,3,2,1,1
3,2,0,0,0,3
```

```
Example Input:
e|-----------------------------------------
B|-----------------------------------------
G|-----------------------------------------
D|--4--7-7-9-9--12-9--12-9--12-9--7--7-9-9-
A|--2--5-5-7-7--10-11-10-11-10-11-5--5-7-7-
E|-----------------------------------------
```

```
Example Output:
E,A,D,G,B,e
-,2,4,-,-,-
-,5,7,-,-,-
-,5,7,-,-,-
-,7,9,-,-,-
-,7,9,-,-,-
-,10,12,-,-,-
-,11,9,-,-,-
-,10,12,-,-,-
-,11,9,-,-,-
-,10,12,-,-,-
-,11,9,-,-,-
-,5,7,-,-,-
-,5,7,-,-,-
-,7,9,-,-,-
-,7,9,-,-,-
```

```
Example Input:
e|-------5-7------7-8-
B|-----5-----5--------
G|---5---------5------
D|-7-------6--------5-
A|--------------------
E|--------------------
```

```
Example Output:
E,A,D,G,B,e
-,-,7,-,-,-
-,-,-,5,-,-
-,-,-,-,5,-
-,-,-,-,-,5
-,-,6,-,-,7
-,-,-,-,5,-
-,-,-,5,-,-
-,-,-,-,-,7
-,-,5,-,-,8
```