The Problem

You work as a developer at ***LinkedOut***, a platform that connects companies with professionals.

LinkedOut operates a website for both professionals and companies to register and seek for jobs and candidates, respectively.

The basic goal of *LinkedOut* is to match professionals to companies so that both parts will be happy.

How professionals and companies choose each other?

1. Each professional compiles a "**professional wish list**" that is a rank-ordered list of companies that she/he wishes to work for.
2. Every company compiles a similar list of professionals in order from most to least preferable to hire. Moreover, each company provides the number of its currently open positions (***slots***) that wishes to fill.

Scoring

Solutions will be scored taking into account the following factors:

1. Penalty points. For each match that **both of the below points are true** the solution gets one penalty point:
    - Professional **P** would prefer to work for company **C** over the company he is currently matched with

    and, simultaneously,
    - Company **C** would prefer professional **P** over one of the candidates currently filling one of its slots.
2. Tiebreaker: Total execution time

Do keep in mind that during the scoring process very large lists from both companies and professionals will try to be matched, so trying to brute force the problem may not get you as far as you wish.

For a scoring example see this case.

Goal

Your project is to develop an algorithm that will match professionals to companies so that the final result is very close to an "ideal hiring".

The final deliverables should be:

1. The executable. A command line program that will take as arguments the paths of two files containing the input and will return as an output the path of a single file containing the answer.

2. The full source code of this program.

Constraints

- In order to have the same reference point for all submissions all executables will run in the same host running Debian Jessie. So, you are welcome to use any language you like***** as long as it produces an executable file that runs on the host. Please see the examples section for the file formats of the input and output files.
- During program execution there will be no internet/network available.
- External libraries are allowed as long as they do not implement the core algorithm of your solution.

* *Please contact me first if you are not going to use one of the following: C# (**.NET Core SDK**), Java 8 (as well as Groovy, Scala, Clojure), C, Python, JavaScript (Node.js v6.x), Ruby or Bash*

More Details

For the sake of simplicity let's assume that every company list includes every single professional and every professional ranks all the companies in the platform.

Example

Program usage

```
usage:
myprog input_companies input_professionals output_file
```

File Formats

input_companies

Description:
free_slots, company_name: comma separated list of employes in order of most to least preferable

Sample:
1, Microsoft: Bill, Jeff, Mark, Larry
1, Amazon: Bill, Jeff, Mark, Larry
1, Facebook: Mark, Jeff, Bill, Larry
1, Oracle: Bill, Mark, Jeff, Larry

input_professionals

Description:
name_of_professional: comma separated list of companies in order of most to least preferable

Sample:
Bill: Microsoft, Facebook, Oracle, Amazon

Jeff: Amazon, Oracle, Microsoft, Facebook
Mark: Facebook, Oracle, Microsoft, Amazon
Larry: Amazon, Oracle, Microsoft, Facebook

Scoring

output_file

Description:
name of company: comma separated list of employes

Sample:
Microsoft: Bill
Amazon: Jeff
Oracle: Larry
Facebook: Mark

Above output is an ideal hiring, so it gets 0 penalty points.

Let's see a non-ideal hiring:

Sample #2:
Microsoft: **Larry**
Amazon: Jeff
Oracle: **Bill**
Facebook: Mark

This output gets ***+2*** penalty points because compared to the ideal one we have two professionals that would prefer to work for another company over the company they are currently matched.

Q&A

Coming soon...