Recruitment - Software Engineer (OpsTech) - Technical challenge

Challenge description

Our business operates several production facilities that produce bouquets. For this technical challenge, we've greatly simplified how the real ones work. At a high level:

- *flowers* of different *species* and *sizes* are used as *input*;
- *bouquets* are produced, according to *design* specifications, as *output*;
- *flowers* arrive at the facility individually and can be stored there until there are enough *flowers* to create a *bouquet*.

Your job is to create a command line application that:

- takes a stream of *design* specifications
- and *flowers* as *input;*
- produces a stream of *bouquets* as *output;*
- produces bouquets *as soon as* enough flowers have been provided to satisfy a design.

In general, you should code the application in the software language that you will be using in the role you're applying from (Python or Ruby). However, you're welcome to write the application in the language you're most familiar with if you would prefer - but it may make it more difficult for our staff to review it.

The solution must have all configuration files needed to be built and run in a Docker container (don't expect anything else but Docker to be installed).

Completing the challenge should take approximately 2-4 hours.

We will evaluate your solution on its correctness as well as its design and overall code quality.

Good luck!

Input/output specifications

- The solution must work with standard input and output (stdin & stdout).
- The ***input*** contains ***designs*** to be produced and a stream of incoming ***flowers***::

```
design1
design2
...
designN
<empty line>
flower1
flower2
flower3
...
```

- The ***output*** should be a ***bouquet*** as soon as one can be created from the available ***flowers***:

```
bouquet1
bouquet2
...
```

Data specifications

- A ***flower species*** is identified by a single, lowercase letter: `a - z.`
- A ***flower size*** is indicated by a single, uppercase letter: `L` (large) and `S` (small).
- A ***flower*** is identified by a ***flower species*** and a ***flower size***: for example, `rL.`
- A ***design name*** is indicated by a single, uppercase letter: `A - Z.`
- A ***design*** is single line of characters with the following format:

```
<design name><flower size><flower1 max quantity><flower1 species>...<flowerN max quantity><flowerN species><total quantity>
```

  - The format includes ***flower* size** only once and it defines the size for all flowers in the given design (i.e. a large *design* can only have large *flowers).*
  - The ***flower species*** are listed in alphabetic order and only appear once***.***
  - The ***flower max quantities*** are always larger than 0***.*** The ***flower min quantities*** are implicit and always equal to 1 (for all specified species).
  - The **total quantity** of flowers can be smaller than the sum of the **flower max quantities** - allowing for some variation between required flower species.
  - Example***:*** `AL1d2r3t5`
- A ***bouquet*** is single line of characters with the following format:

```
<design name><flower size><flower1 quantity><flower1 species>...<flowerN quantity><flowerN species>
```

  - The format includes ***flower* size** only once and it defines the size of all flowers in the given bouquet (i.e. a large *bouquet* can only have large *flowers).*
  - The ***flower species*** are listed in alphabetic order and only appear once***.***
  - The ***flower quantities*** are always larger than 0***.***
  - Example***:*** `AL1d2r2t`
- `A bouquet must comply to its design:`
  - A ***bouquet*** must have all and only ***flower species*** required by the corresponding ***design*** (i.e. comply with the implicit ***flower min quantities***)***.***
  - Every required **flower *species*** in a ***bouquet*** must be in the ***flower quantity*** that is less or equal to the ***flower max quantity*** specified by the ***design***.
  - The sum of the ***flower quantities*** in a ***bouquet*** should be equal to the ***total quantity*** of flowers in the corresponding ***design.***

Example

The following ***input***

```
AS2a2b3
BL2a2

aL
bS
aS
bS
aS
aL
aS
bS
```

should produce the following ***output***

```
AS1a2b
BL2a
AS2a1b
```

Questions?

In case things aren't clear enough and/or not explicitly specified - please use your best judgment (but keep it simple). And don't forget to mention those in the readme!

Wrap up

Are you done? Great!! Please submit your solution in a private GitHub repository and grant access to "BloomAndWildReviewer" user.

Then, email us to let us know that you’ve done this step, and are ready for review.

Thank you for participating in our code challenge!