A bookseller has lots of books classified in 26 categories labeled A, B, ... Z. Each book has a code c of 3, 4, 5 or more capitals letters. The 1st letter of a code is the capital letter of the book category. In the bookseller's stocklist each code c is followed by a space and by a positive integer n (int n >= 0) which indicates the quantity of books of this code in stock.

For example an extract of one of the stocklists could be:

```
L = {"ABART 20", "CDXEF 50", "BKWRK 25", "BTSQZ 89", "DRTYM 60"}.
```

or

```
L = ["ABART 20", "CDXEF 50", "BKWRK 25", "BTSQZ 89", "DRTYM 60"] or ....
```

You will be given a stocklist (e.g. : L) and a list of categories in capital letters e.g :

```
M = {"A", "B", "C", "W"}
```

or

```
M = ["A", "B", "C", "W"] or ...
```

and your task is to find all the books of L with codes belonging to each category of M and to sum their quantity according to each category.

For the lists L and M of example you have to return the string (in Haskell/Clojure/Racket a list of pairs):

```
(A : 20) - (B : 114) - (C : 50) - (W : 0)
```

where A, B, C, W are the categories, 20 is the sum of the unique book of category A, 114 the sum corresponding to "BKWRK" and "BTSQZ", 50 corresponding to "CDXEF" and 0 to category 'W' since there are no code beginning with W.

If L or M are empty return string is `""` (Clojure and Racket should return an empty array/list instead).

## Note:

In the result codes and their values are in the same order as in M.