K-Means

It is Halloween, and you need to figure out how best to manage your various candy-related tasks! Which candy is best? Which is most popular with visitors? What candy should you give out to visitors, and what candy should you hoard for yourself!? All of these questions flood your mind, and it becomes difficult to rationalize how best to optimize your candy experience.

After thinking for a bit, you determine that it is best to have a good variety of candy. You want to buy a good mix of sweet, sour, expensive, cheap, small, and large candy for Halloween. You decide that the best course of action is to write a program that will look at different types of candy and assign them into groups that are all similar to one another (that way, you can make sure to purchase at least one candy from each group).

Given a list of candy, a list of features / properties of those candy, and the number of groups to create, can you find the groups of candy that are most similar to one another? You **MUST** do this by using the **k-means** algorithm as discussed in class.

Input

There will only be one test case per input file. The first line of input will contain integers F, K, and S which represent the number of features of each candy measured, the number of clusters we want to establish, and the number of iterations the algorithm should run respectively. Your solution **must use the k-means algorithm** and update the cluster locations exactly S times.

The next line of input will contain exactly F strings, representing the names of the candy features that were measured (e.g., price, sweet, size, etc.).

In order for the algorithm to produce consistent results, the starting locations of the clusters must be fixed. The next K lines of input each contain F double values per line. These represent where the K cluster locations should be initialized.

The next line of input contains an integer T, the number of training examples provided. The next T lines after that contain a string (representing the name of the candy) followed by exactly F double values representing the measurements of each feature on that candy.

Output

For each cluster, you should output its final location (F doubles) to four decimal places. Below the cluster location, print every candy that was grouped with that cluster. You should print out the clusters in the same relative order they were originally provided to you in the input. In addition, all candy should be printed in the relative order it was given in the input file.

Submission

You should submit your code in Java, C++, or Python. You should configure your makefile properly to run your code.

Sample Input

```
3 2 20
price sweet size
0.0 0.0 0.0
2.0 2.0 2.0
8
Smarties 1.0 11.0 5.0
Nerds 1.3 12.0 5.0
DumDums 1.5 9.0 2.1
Twix 6.2 2.1 5.8
Snickers 6.1 2.0 5.1
SweetTarts 1.9 15.0 4.12
AirHeadsExtreme 1.1 21.0 5.2
DarkChocolate 7.0 1.1 3.9
```

Sample Output

```
6.4333 1.7333 4.9333
Twix
Snickers
DarkChocolate
1.3600 13.6000 4.2840
Smarties
Nerds
DumDums
SweetTarts
AirHeadsExtreme
```