In this problem, you're to implement a simple recommendation system. There are n users, each of them rated some of the m movies.

For example, $n = 7$, $m = 6$, the ratings are shown in the following table:

	M1	M2	M3	M4	M5	M6
U1	2.5	3.5	3.0	3.5	2.5	3.0
U2	3.0	3.5	1.5	5.0	3.5	3.0
U3	2.5	3.0		3.5		4.0
U4		3.5	3.0	4.0	2.5	4.5
U5	3.0	4.0	2.0	3.0	2.0	3.0
U6	3.0	4.0		5.0	3.5	3.0
U7		4.5		4.0	1.0	

We can see that there are 3 movies that user 7 haven't watched: M1, M3 and M6.

Question: Which one do we recommend?

One of the most popular methods to solve this is collaborative filtering. For example, we can:

- Step 1: Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
- Step 2: Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user.

This is known as user-based collaborative filtering. Alternatively, item-based collaborative filtering invented by Amazon.com (users who bought x also bought y), proceeds in an item-centric manner:

- Step 1: Build an item-item matrix determining relationships between pairs of items.
- Step 2: Using the matrix, and the data on the current user, infer his taste.

This is what we use in this problem.

Building the Matrix

The similarity matrix has m rows and m columns. The element in row i and column j, denoted by $S(i, j)$, is the similarity of movie i and movie j. To calculate $S(i, j)$, we first calculate the "rating difference" for each user who rating both movie i and movie j, then let x be the sum of the squares of these differences, then $S(i, j) = 1/(1 + x)$.

For example, to calculate $S(2, 3)$, we first find out the rating differences: $|3.5-3.0| = 0.5$, $|3.5-1.5| = 2.0$, $|3.5-3.0| = 0.5$, $|4.0-2.0| = 2.0$, then $x = 0.5^2+2.0^2+0.5^2+2.0^2 = 8.5$, so $S_{2,3} = 1/(1+8.5) = 0.105$.

The complete similarity matrix of the example above is (the matrix is symmetric so we omit some elements):

	M1	M2	M3	M4	M5	M6
M1	1	0.222	0.222	0.091	0.400	0.286
M2		1	0.105	0.167	0.051	0.182
M3			1	0.065	0.182	0.154
M4				1	0.053	0.103
M5					1	0.148
M6						1

Making Recommendations

Once we have the similarity matrix, it's easy to make recommendations. Suppose we want to make recommendations for user u, then the (predicted) score for each "unwatched" movie is simply the *weighted average* of his ratings of the "watched" movies.

To be more specific, suppose user u has watched k movies m_1, m_2, ..., m_k, then the score of an unwatched movie i equals to:

$$(S(i, m_1) * rating(m_1) + \ldots + S(i, m_k) * rating(m_k))/(S(i, m_1) + \ldots + S(i, m_k))$$

For example, the score of movie 6 for user 7 is:

$$(S(6, 2) * 4.5 + S(6, 4) * 4.0 + S(6, 5) * 1.0)/(S(6, 2) + S(6, 4) + S(6, 5)) = 3.183$$

Actually, this score is higher than movie 1 and movie 3, so we recommend movie 6 to user 7.

Note that in the formula above, if the denominator is zero, which means movie i is not at all similar to any movie he watched, we should not recommend this movie.

Input

There will be only a single test case. The first line contains 3 integers, n, m and c ($1 \le n \le 50$, $1 \le m \le 200$, $1 \le c \le nm$). Each of the next c lines contains two integers i, j ($1 \le i \le n$, $1 \le j \le m$), and a real number r between 0 and 5, that means user i's rating of movie j is r. Nobody will rate the same movie twice. Then there will be several lines, each containing an integer u. That means we need to recommend 10 movies to user u. Every user has a least one unwatched movie that is somewhat similar to his watched movie.

Output

For each request, print the top 10 recommended movies: the movie number, and the score (to three digits after the decimal point), in descending order to score. The input is carefully designed so that the final output will not be changed due to floating-point errors. If there are less than 10 unwatched movies that are somewhat similar to his watched movies, simply display them all (but still sorted). Print a blank line after each user.

Sample Input

```
7 6 35
1 1 2.5
1 2 3.5
1 3 3.0
1 4 3.5
1 5 2.5
1 6 3.0
2 1 3.0
2 2 3.5
2 3 1.5
2 4 5.0
2 6 3.0
2 5 3.5
3 1 2.5
3 2 3.0
3 4 3.5
3 6 4.0
4 2 3.5
4 3 3.0
4 6 4.5
4 4 4.0
4 5 2.5
5 1 3.0
5 2 4.0
5 3 2.0
5 4 3.0
5 6 3.0
5 5 2.0
6 1 3.0
6 2 4.0
6 6 3.0
6 4 5.0
6 5 3.5
7 2 4.5
7 5 1.0
7 4 4.0
7
7
```

Sample Output

```
Recommendations for user 7:
6 3.183
3 2.598
1 2.473

Recommendations for user 7:
6 3.183
3 2.598
1 2.473
```