Problem F. Finding Suspicious Proteins

Source file name:	Finding.c, Finding.cpp, Finding.java, Finding.py
Input:	`Standard`
Output:	`Standard`

Little Claire studies proteins, which are sequences of amino acids. There are 20 amino acids from which proteins are built. While amino acids all have proper names, such as *alanine* or *glycine*, they are often denoted by single letters, so that proteins can be seen as sequences of different lengths, such as `DTASDAAAAAALTAABAAAAAKLTABBAAAAAAAATAA`, `TIFLQQQQQQQQQQ` or even maybe `RICKRQLL`.

Comparing two proteins can be difficult, because they may contain active sites, which determine their function in a cell, and less important parts of the sequence. Recent advances in artificial neural networks made it possible to train a network that, given a protein, outputs a sequence of l numbers, where each number roughly corresponds to a feature of a protein that correlates with its possible functions in a cell. Such a sequence is called an *embedding*.

Claire is particularly interested in *suspicious* proteins, those which are really different from others. For this purpose, she considers the so-called *Manhattan distance* between embeddings of proteins. For two protein embeddings p and q of length l, the distance $\mathcal{D}(p, q)$ is computed as follows:

$$\mathcal{D}(p, q) = \sum_{i=1}^{l} |p_i - q_i|,$$

where p_i is the i-th element of the embedding p.

Claire wants to find k suspicious proteins in the given list of n proteins. As a baseline for her studies, Claire wants to use the following greedy algorithm:

- Find a protein $p^{(1)}$ which is the most distant from the first protein in the list.
- The second protein, $p^{(2)}$, is chosen as the most distant protein from $p^{(1)}$.
- The third one, $p^{(3)}$, is chosen so that $\min\{\mathcal{D}\left(p^{(1)}, p^{(3)}\right), \mathcal{D}\left(p^{(2)}, p^{(3)}\right)\}$ is maximum possible. That is, it must be far away from *both* previously chosen proteins.
- All subsequent proteins $p^{(i)}$, $4 \leq i \leq k$, are chosen in a similar way: the minimum of the distances to all the previously chosen proteins should be maximum possible.

Note that, in the case of ties, the first matching protein in the list must be chosen.

Claire's implementation works nicely for small numbers n and k, but becomes too slow as they increase. You must find a way to optimise this.

Input

The first line contains three numbers n ($3 \leq n \leq 10^4$), l ($1 \leq l \leq 100$) and k ($2 \leq k \leq \min\{n, 256\}$): the overall number of proteins, the length of each protein embedding, and the number of proteins to choose.

Each of the following n lines starts with a protein identifier, which is a sequence of at least one and most ten capital letters and/or numbers. Then, separated by whitespace, come l single-digit integer numbers $v_{1\ldots l}$ ($0 \leq v \leq 9$), which define the embedding of the protein. All protein identifiers will be different.

Output

Output the identifiers of k chosen proteins, one per line, in their respective order ($p^{(1)}$ to $p^{(k)}$).

Example

Input	Output
4 2 2 FIRST 3 4 SECOND 1 2 THIRD 8 7 FOURTH 5 6	THIRD SECOND
6 5 3 1OGLOBIN 1 1 1 1 1 GLU1O 9 9 9 9 9 8EIN 8 9 8 9 9 COLLA6EN 6 5 4 3 2 7ILK 3 4 5 6 7 OLBUMIN 1 2 0 2 1	GLU1O 1OGLOBIN 7ILK