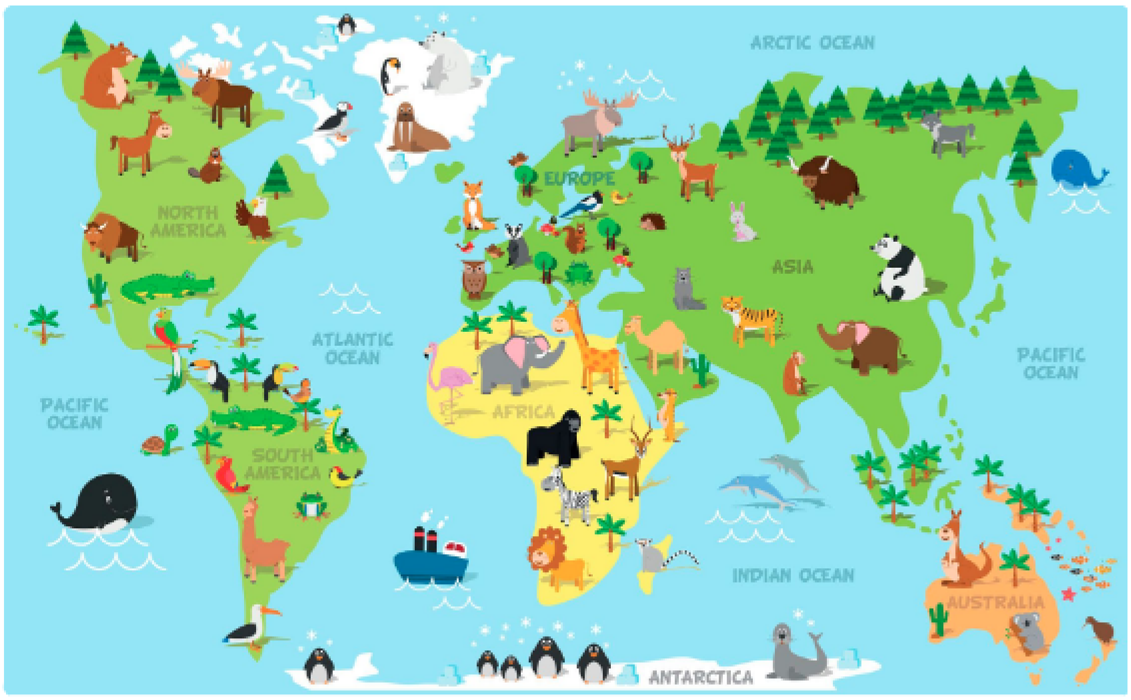
Map

Your kooky best friend recently decided to quit his job and fly the world in a hot air balloon. You're obviously worried, but since you couldn't convince him to stay, you bought him a smartphone with a SkyLink connection and asked him to text you any chance he gets.
Days have passed and he hasn't texted, but he's been taking pretty pictures from his balloon and posting stories on Instagram.
Well, you figure, that should be enough info to locate him accurately.

Data

You're given a very accurate map of the world (Figure 1) and a set of small patches from this map. Some patches might look like the original map (#nofilter), some might have (corny, no offense) filters on them. Your task is to find the position of the top-left corner of each patch in the map.



Side note: Earth is, as we all know, flat and rectangular.

There are two data sets:

- *Public data set* is used for developing your solution. After you submit your solution, you will be able to see how well your solution performs against this data set. *Public data set* is not used for calculating the final score. Public data set is available **here**.
- *Private data set* is used for testing your solution. The final score will be measured against this data set. *Private data set* and the final score will be available after the homework finishes. *Private data set* contains different data than the *public data set*, but the world map will be the same for every test case in both sets, and distribution will be similar.

Implementation details

Input

Inputs are given through standard input.

First line contains the path to world map file in PNG format (it will always be the exact copy of the image provided in Figure 1)

Second line contains the number of patches in the test sample, N.

Third line contains two integers separated by space, the patch size "width height" in pixels. Each patch in this test sample will have the same size.

Each of the next N lines contains a path to an image patch (in PNG format).

Example of the input:

```
D:\data\public\set\map.png
4
40 40
D:\data\public\set\0\0.png
D:\data\public\set\0\1.png
D:\data\public\set\0\2.png
D:\data\public\set\0\3.png
```

Output

All results should be printed to the standard output.

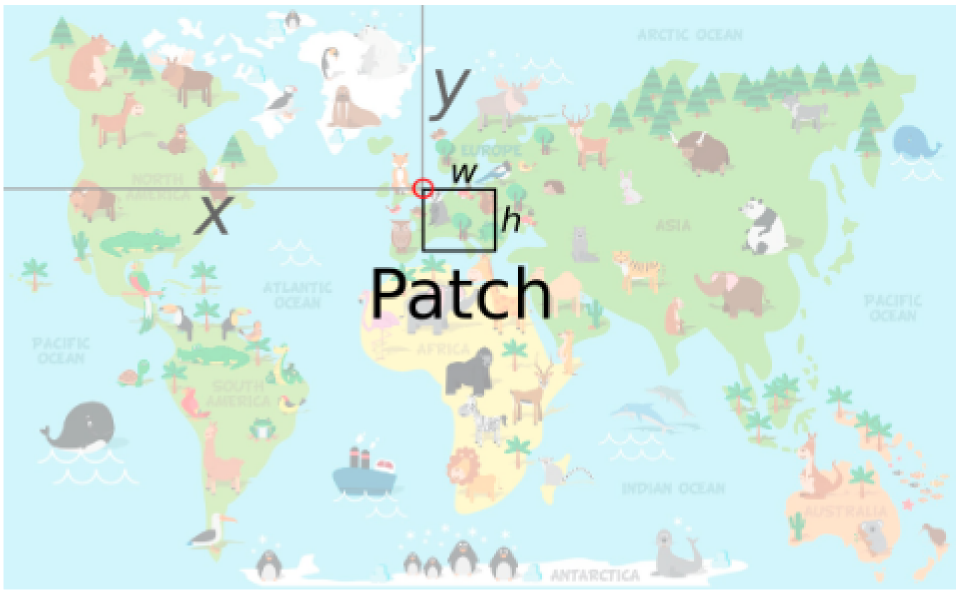
Your solution should contain N lines, each containing two integers, X and Y coordinates separated by comma (,) of the top-left corner of the patch in the world map (see Figure 2 for clarity).

Example of the output:

```
240,189
6,113
420,225
102,221
```

Notes:

- even if you're not able to accurately pinpoint your friend, you should provide the best estimate you can.
- you need to give output for every patch in test case (not less than that, and not more than that), otherwise you'll get 0 points for test case.



Available packages

- `numpy`
- `scipy`
- all packages from the standard python library

Scoring

For each patch, you will receive `max((1 - (D / 100)^2), 0)` points, where `D` is the Euclidean distance (in pixels) between your solution and the correct solution. This means that you'll receive 1 point for the exact solution, and fractions of a point for any solution within 100 pixels from the exact solution (if distance is larger than 100 pixels, you receive 0). Score for each of N patches in test case will be summed and scaled to a total of 40 points for test case, following formula `(sum_of_scores / N) * 40` rounded to integer.

For example, if test case has 50 patches, and you got 1 on 25 of them, and 0 on rest of them, your total score for testcase will be `(25 / 50) * 40 = 20`.

Constraints

- Time limit is 20s.
- Memory limit is 1GB.

If in doubt, please refer to the data from *public data set* and proceed with a reasonable assumption.