A. Placing Rectangles

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a rectangular region P of coordinate plane (limited by straight lines $x = 0$, $y = 0$, $x = w$ and $y = h$). Your task is to place n given rectangles or at least some of them on the plane. The rectangles are given with their sizes: w_i and h_i are the width (size along the Ox axis) and the height (size along the Oy axis) of the i-th rectangle. When placing rectangles on the region P, you are allowed to rotate them by 90 degrees and change the scale: you can stretch the rectangles as well as shrink them. When you change the scale, you must preserve the rectangles' proportions. The rectangles can be scaled in the following limits (in linear sizes): Each rectangle can not be increased by more than two times, or reduced by more than 10 times. The rectangles should be placed so that no two rectangles intersect internally, that is, share a positive area. The rectangles can touch each other.

Your program will be awarded points based on the location you find. Let's consider all pairs of touching rectangles and look at the length of their common segment (let it be equal to l_{ij}).

- If two rectangles have the same orientation (i.e., both have not been rotated, or both have been rotated relatively to its initial position), then l_{ij} is substracted from the score.
- If a pair of rectangles has a different orientation (i.e., one has been rotated relatively to its initial position, and the other hasn't), then l_{ij} is added to the score.

The number of points earned on all pairs of the touching rectangles will be the result of the program's working on the test.
Write a program that will position the rectangles in the region P so as to maximize the number of earned score points. You should scale and position the rectangles so that the coordinates of any corner after multiplying them by 10 were integers. It is not necessary to place all rectangles on the field, some of them can be discarded.

Your program will be launched on the tests, which are generated by program `gen` (see on http://code.google.com/p/vkcup-2012-wildcard-round2-toolbox/). The output of your program can be evaluated using the `check` program from the same archive. To run it, it is enough to run "`check input-file output-file`".

Generator `gen` was launched nine times with random arguments to get pretests 2-10. The first pretest (which also is the first sample) is made manually. Tests from the statement usually match the first pretests. During the contests, all solutions will be tested only on these pretests. You do not know the pretests (except for the samples), but a performance report will be available to you, showing the performance on each test with some information about it and the result of your program. The current table of results will be based on the last solution attempt for each participant. For each participant, we calculate the number of points earned on the pretests, and the participants are listed in the order of non-increasing of this value. The attempts that ended with an unsuccessful run on all pretests (compilation errors, run-time errors on all tests, etc.) will be ignored.

After the end of the main time of the contest, for each participant we will select his last non-ignored attempt and re-test it on the full set of tests. The main set of tests will be obtained in a similar way, it will contain 500 tests. It will not contain the pretests (to be specific, the samples).

The official final results will be ordered by the sum of scores for all tests of the main set.

The jury warns you that during the contest the archive of tools may be updated, the programs that form it may be changed or expanded. Some clarification of the statement or changing some details there is also possible. Pay attention to changes. All changes will be reflected in the text of the problem statement.

During the contest is strictly forbidden to post/discuss algorithms/approaches for the problem, share any conclusions about the problem. You can not share the results (including just report score points) of the solutions on any tests. It is prohibited to publish tools to simplify and automate the process of solving the problem.

UPD: Minor fix done in visualizator. Archive version 1.0.2 is available.

Input

The first line of the input contains two integers w and h $(10 \le w, h \le 100)$ — the size of the given rectangle area. The second line contains integer n $(5 \le n \le 100)$ — the number of rectangles to be placed.

The following n lines contain descriptions of the rectangles, one per line. Each rectangle is specified by a pair of integers w_i, h_i $(1 \le w_i \le w, 1 \le h_i \le h)$ — its width and height, respectively. It is guaranteed that $w_i \ne h_i, \le w_i, \le h_i$. In addition, the given rectangles aren't too long, namely $0.1 \le \ \le 10$.

All tests were generated with programme `gen` from the archive by the link above. To generate a test just run "`gen some-random-token`"

Output

Print n lines. On the i-th line describe the i-th rectangle's position. Print "`-1 -1 -1 -1`" (without the quotes), if the rectangle isn't positioned in the area. Otherwise, print four coordinates of its corners "$x_1\ y_1\ x_2\ y_2$" (without the quotes) with no more that one digit after the decimal point.

Examples

input

```
30 40
5
5 10
12 4
10 11
6 10
4 9
```

output

```
27.5 0.0 30.0 5.0
-1 -1 -1 -1
15.5 0.0 25.5 11.0
7.4 11.0 27.4 23.0
3.4 11.0 7.4 20.0
```

input

```
41 48
34
36 7
15 26
33 28
31 5
29 10
5 39
5 43
35 8
21 13
8 43
7 27
34 7
19 10
9 39
29 17
12 38
30 14
9 5
41 36
41 25
31 37
31 8
32 29
7 39
21 9
```

```
6 35
31 9
7 43
36 5
20 39
18 8
14 5
8 32
8 39
```

output

```
Not available
```

input

```
72 84
29
9 78
51 10
10 77
14 81
18 60
27 13
10 16
21 35
29 69
12 80
22 67
58 40
41 19
11 47
25 64
10 28
9 60
8 52
40 71
15 25
41 12
11 59
11 83
8 36
8 48
27 55
9 58
27 52
61 20
```

output

```
Not available
```