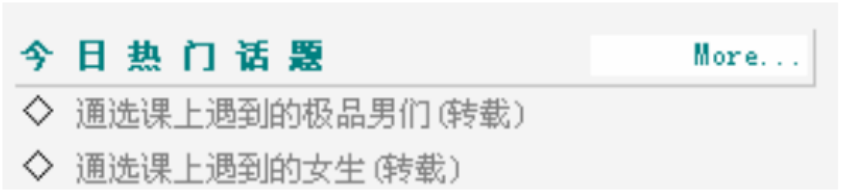
4755 Seat taking up is tough

Students often have problems taking up seats. When two students want the same seat, a quarrel will probably begin.



It will have very bad effect when such subjects occur on the BBS.

So, we urgently need a seat-taking-up rule. After several days of argument, the rule finally comes out:

As shown in the figure on the right, the seats in a classroom form a $n \times m$ grid (n rows and m columns), and every cell in the grid represents a seat. The coordinates of the seat in the north-west corner are (1,1) and the coordinates of the seat in the south-east corner seat are (n, m). As you know, some seats make you feel good and some seats don't. So every seat has a "feeling index".

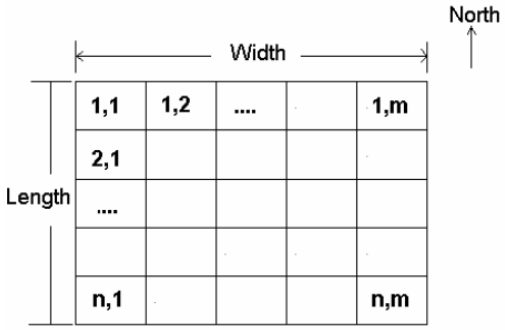


Figure 1. A class room

Students can take up seats for himself and his friends. Of course, if a seat is already taken up by a student, it can't be taken up by others.

For the convenience of communication between friends, when a student is trying to take up seats, he wants all the seats he needs to be consecutive and in the same row. If he can do that, he takes up all the seats he needs, and save the most western one for himself. For example, if a student wants to take up 3 seats, then taking (2,2),(2,3),(2,4) and saving (2,2) for himself is ok; but taking (2,2), (2,4),(2,5) is invalid because those seats are not consecutive. Under the precondition of accomplishing his seat-taking job, a student always wants the "feeling index" of his seat to be as large as possible.

However, if a student cannot take up all the seats he needs, he will just try to take up only one seat for himself because he doesn't want to get into the trouble of explaining "Why they can get seats but I can't?" to some of his friends. Of course he still wants the "feeling index" of his seat to be as large as possible in that situation.

Everyone wants to know where are the seats he can take up. This problem seems a little bit complicated for them. So they want you to write a program to solve the problem.

Input

There are several test cases and the input ended by a line of '`0 0 0`'.

For each test case:

The first line contains three integers: n, m and k ($1 \le n, m \le 30$, $1 \le k \le 50$). It means that there are n rows of seats in the classroom and there are m seats in every row. k is the number of students who come into the classroom trying to take up seats.

Following are n lines describing the seats by north to south order. Each line represents a row of seats and contains m integers, indicating the "feeling index" of every seat in that row, by west to east order. "Feeling index" can be fit in a 32-bit integer.

Then k lines follow (We call them k "student lines"). Each line is in the format of 'hh:mm q' ($0 \leq hh < 24$, $0 \leq mm \leq 59$, $1 \leq q \leq 50$) meaning that a student comes into the classroom at mm minutes past hh o'clock, trying to take up q seats. mm and hh are all two digit integers with possible leading zero, and q is also an integer. Please note that when a student enters the class room, he begins to do his seat taking job immediately and the job takes no time.

It is guaranteed that the "feeling index" of every seat is different and no students come into the classroom at the same time.

Output

You should output k lines for each test case, in the order that correspondent to the above mentioned k "student lines" in the input. Each line must contain two integers indicating the coordinates of the seat which is saved by the student for himself. If the student can't take up any seats, just output a '`-1`' instead.

Sample Input

```
5 5 8
11 12 15 14 13
21 22 25 24 23
16 17 20 19 18
6 7 10 8 9
1 2 5 4 3
09:00 2
09:01 5
09:02 5
09:03 5
09:04 5
09:05 3
09:06 2
09:07 3
0 0 0
```

Sample Output

```
2 3
3 1
1 1
4 1
5 1
2 5
2 1
-1
```