A. Scheduler for Invokers

time limit per test: 30 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

In this problem you need to write job distribution subsystem for testing submissions.

Invokers (simplistically) are components of testing system, which test given submission on single test and give back a verdict. In this problem two verdicts are possible — OK (test passed) or RJ (rejected, test failed).

Job distribution subsystem will be named *scheduler*. You have to implement it.

For each problem two parameters are given: number of tests in a problem and time limit for a problem.

We will consider that the system works discreetly and one tick equals 10 milliseconds. If some event occupied in time moment not divisible by 10 ms then scheduler will get it in the nearest next tick.

After the end of the contest (i.e., a week after its start) the last solution you sent (having positive score) will be chosen to be launched on the final tests. The final tests are confidential and distinct from those that will be used during the competition. The total number of points that will be scored in the final tests will determine the winner of the competition.

Specially for this round we implemented feature to submit ZIP-archive with multiple source files. All files should be in root of ZIP-file, no directories are allowed. You can use this feature for:

- Java 8: `main` should be in a class `Main` of default package;
- GNU C++ 11: exactly one file should contain the entry point `main`, all files to compile should have extension `cpp`.

You will be receiving submissions and invoker verdicts in interactive mode. Please be sure to use the stream flushing operation after each of your moves to prevent output buffering. For example, you can use "`fflush(stdout)`" in C or C++, "`System.out.flush()`" in Java and "`flush(output)`" in Pascal.

The problem is estimated as follows: let a_i is a full testing time (including expectations) of the i-th submission, that is time, passed from the moment of receiving the submission until the moment when it is fully tested. Let's find $r = \sqrt[n]{\sum_{i=0}^{n} a_i}$, where n is the number of submissions to test. The exact number of points for single test is calculated by the formula $\sqrt{A/B} \cdot 100$, where A is a r for some simple scheduler, written by jury, and B is a r for your *scheduler*.

Materials for local testing will be published soon. The package with materials will contain an interactor and a set of test data for the interactor.

Input

On the start your program reads the following data from standard input:

- in the first line you are given an integer number t ($1 \le t \le 500$) — the number of available invokers (all invokers work simultaneously and independently, free invoker starts testing a job as soon as it is appointed to him);
- in the second line you are given an integer number p ($1 \le p \le 10000$) — the number of problems, submissions to which are expected to be tested.

Further p lines with description of problems are followed. Each description contains problem time limit (integer number between 250 and 30000) and the number of tests (from 1 to 1000).

The following data is given in interactive mode. It means that a following block of data will be available only after your *scheduler* writes information of its behavior on the previous tick. Firstly, block of data about new submissions, needed to be tested, follows for a tick. Each submission is described by single integer number — index of the problem (from 0 to $p - 1$). The value `-1` means that there are no new submissions, needed to be tested, in this tick.

Further block of testing results (from invokers) in this tick follows. These lines contain three elements: index of submission, a number of test and the verdict of test (OK or RJ). The block of testing results ends with a line "`-1 -1`".

Consider that problems and tests are numbered (from zero) in the order of its appearance. Total number of submissions for one test does not exceed 20000.

Output

After reading the data about the next tick your scheduler can decide to send submissions for exact testing. The submission can be tested in any test at any time. Your scheduler must output a couple of integers — index of submission and a number of test in a single line in order to start a test for the submission. If there are a free invoker, a testing of a submission immediately begins

(invoker becomes busy until the result comes back). Otherwise, your request will be ignored. When you are done sending requests, output line "-1 -1".

You can simply maintain the amount of free invokers in your program, reducing each time variable t while sending the task to invokers (if $t = 0$, there is no need to send) and increasing t, if invoker returns a verdict.

It is considered that the submission was tested if it was tested on all tests of the problem, or if it was tested on all tests up to first RJ verdict (inclusive).

Interactor breaks out at a time when all the planned submissions will be tested completely. In this case your submission should stop, as soon as it finds closing of input data stream.

Example

input

```
1
1
500 2
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
0
-1
-1 -1
0
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
0 0 OK
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
0 1 OK
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
-1 -1
-1
1 0 RJ
-1 -1
```

output

```
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
0 0
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
0 1
-1 -1
-1 -1
-1 -1
-1 -1
```

```
-1 -1
-1 -1
1 0
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
-1 -1
1 1
-1 -1
```

Note

Materials and tests are available now at http://assets.codeforces.com/files/vk/vkcup-2016-wr2-materials-v1.tar.gz. Please read README.txt to learn how to test your solution with the interactor.