Code With No Forces

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Preparing a reliable programming contest is tricky work. Sometimes, you can make the test set huge (e.g., `wrong answer on test 400`) instead of guessing how participants will solve the problems. However, huge test sets could be a critical burden to the poorly implemented contest platforms. If the judging process is not parallelled well, the flow time of submission is unacceptable, and you will receive countless complaints about it.

The problem you've made contains n test cases, and there are m test solutions written by different testers that you can believe are the most typical solutions. To avoid long judging processes, you want to prune the test set by deleting some test cases and reordering the remaining if needed. After that, it's required to keep the results of every test solution remain unchanged. Your goal is to keep the least number of tests at last.

Take the following snapshot from the contest preparing platform, Polygon, as an example.

#	aho_tle.cpp	hash_1e18.cpp	hash_1e9.cpp	hash_kmp.cpp	hash_overflow.cpp	std.cpp
1	OK 0 / 34	OK 15 / 1	OK 0 / 1	OK 0 / 4	OK 0 / 1	OK 0 / 36
2	OK 0 / 34	OK 0 / 1	OK 15 / 1	OK 15 / 4	OK 0 / 1	OK 0 / 36
3	OK 0 / 34	OK 15 / 1	OK 0 / 1	OK 0 / 4	OK 15 / 1	OK 0 / 36
4	OK 0 / 34	OK 0 / 1	OK 0 / 1	OK 0 / 4	OK 15 / 1	OK 0 / 36
5	OK 0 / 34	OK 0 / 1	OK 15 / 1	OK 0 / 4	OK 0 / 1	OK 0 / 36
6	OK 0 / 34	OK 0 / 1	OK 0 / 1	OK 0 / 4	OK 0 / 1	OK 0 / 36
7	OK 0 / 34	OK 0 / 1	OK 0 / 1	OK 15 / 4	OK 0 / 1	OK 0 / 36
8	OK 46 / 34	OK 78 / 1	OK 78 / 1	OK 234 / 4	OK 30 / 1	OK 31 / 36
9	OK 31 / 35	OK 592 / 1	OK 499 / 1	OK 389 / 4	OK 390 / 1	OK 46 / 36
10	OK 93 / 34	OK 265 / 1	OK 217 / 1	OK 249 / 4	WA 140 / 1	OK 77 / 36
11	OK 93 / 34	OK 265 / 1	OK 233 / 1	OK 249 / 4	WA 140 / 1	OK 62 / 36
12	OK 779 / 34	OK 2932 / 1	OK 2433 / 1	OK 1730 / 399	OK 2089 / 1	OK 561 / 36
13	OK 811 / 34	OK 3119 / 1	OK 2589 / 1	OK 1809 / 397	OK 2245 / 1	OK 514 / 36
14	OK 811 / 34	OK 2964 / 1	OK 2308 / 1	OK 1731 / 390	OK 2028 / 1	OK 483 / 36
15	OK 109 / 35	OK 296 / 4	OK 249 / 4	OK 312 / 6	OK 140 / 3	OK 78 / 36
16	OK 78 / 35	OK 389 / 2	OK 373 / 2	OK 358 / 5	OK 280 / 2	OK 93 / 36
17	TL 5000 / 34	OK 140 / 1	OK 109 / 1	OK 280 / 6	OK 62 / 1	OK 78 / 36
18	OK 93 / 35	OK 171 / 3	OK 155 / 3	OK 280 / 6	OK 124 / 3	OK 124 / 36
19	OK 93 / 35	OK 202 / 3	OK 202 / 3	OK 327 / 6	OK 93 / 3	OK 109 / 36
20	OK 124 / 35	OK 124 / 2	WA 124 / 2	OK 265 / 5	OK 77 / 2	OK 109 / 36
21	OK 124 / 35	OK 155 / 2	WA 156 / 2	OK 249 / 5	OK 78 / 2	OK 140 / 36
Σ passed tests	20	21	19	21	19	21

Test solutions for *Last Warning Of Competition Finance Officer*, from the 2022 Shanghai Collegiate Programming Contest

The input data (See section **Input**) is given as the sheet above. For each solution, it will have a result with its verdict, running time and memory used on each test. The verdicts you should care about include the following, which are all common verdicts except **compile error**.

- `OK`: **correct**
- `WA`: **wrong answer**
- `TL`: **time limit exceeded**
- `ML`: **memory limit exceeded**
- `RE`: **runtime error**

After testing on a specific data set, the result including a representing verdict, a peak running time and a peak memory used will be returned to the participant. However, participants will not have information after the first failed test:

- When a solution passes all tests (**correct** on all test cases), the maximum time and memory will be shown with the **correct** verdict.
- Otherwise, the first failed test's verdict decides the verdict of the solution, and the maximum time and memory from the first test to the first failed test will be shown.

We can conclude from the above sheet that the verdicts of each solution are shown as follows.

Test Solution	Result
aho_tle.cpp	**time limit exceeded** 5000ms/35MB
hash_1e18.cpp	**correct** 3119ms/4MB
hash_1e9.cpp	**wrong answer** 2589ms/4MB
hash_kmp.cpp	**correct** 1809ms/399MB
hash_overflow.cpp	**wrong answer** 390ms/1MB
std.cpp	**correct** 561ms/36MB

To reduce the size of the test set, you can delete some tests, reorder the remaining ones, and make the results (verdicts, time, memory) of all tester solutions unchanged. Take the above sheet as an example:

- You can keep cases $9, 17, 10, 15, 13, 20, 12$ to satisfy the requirement, and it can be shown to be an optimal solution.
- You can reorder the cases if you like to – $9, 17, 10, 15, 13, 12, 20$ is also valid. However, swapping 9 and 17 is not valid since the solution 'aho_tle.cpp' will have a smaller memory cost.

You are given a test result sheet. Find an optimal way to reduce the test set so that the number of tests remaining is minimum.

Note that if a program is not tested on any test case, the status is undefined and not equal to any valid result, instead of '**correct** 0ms/0MB' on some online judges.

Input

The first line contains two integers n and m ($1 \le n \le 400,\ 1 \le m \le 6$), denoting the size of the test set and the number of tester solutions.

In the following n lines, the i-th line contains m strings separated by spaces, denoting the verdicts of the m test solutions of the i-th tests. The string is in the format '`{verdict},{time cost}/{memory cost}`', where the time cost and memory cost are integers, in milliseconds and megabytes, respectively.

It's guaranteed that:

- `verdict` $\in \{$`OK`, `WA`, `TL`, `ML`, `RE`$\}$.
- $0 \le$ `time cost`, `memory cost` $\le 10\,000$.
- All results with `verdict` = `TL` has an equal maximum value of `time cost` that is higher than those with different verdicts.
- All results with `verdict` = `ML` has an equal maximum value of `memory cost` that is higher than those with different verdicts.

Output

In the first line, print a single integer $|S|$ denoting the minimum size of the test set satisfying the requirements mentioned above. Note that for a correct answer, $1 \le |S| \le n$ must hold.

In the second line, print $|S|$ integers separated by spaces, denoting the indices sequence S of tests that the new test set you constructed is in such order.

Examples

standard input	standard output
2 3 OK,1/1 OK,2/1 OK,2/2 WA,1/1 OK,1/1 TL,1000/1	2 1 2
3 3 OK,1/1 OK,2/1 OK,1/2 OK,3/3 OK,1/2 OK,114/514 WA,999/999 TL,3000/2 ML,999/1024	1 3
5 3 OK,0/0 OK,0/0 OK,0/0 WA,1/0 RE,0/0 OK,0/0 WA,0/0 WA,0/0 WA,0/0 OK,1/0 RE,0/0 OK,0/0 WA,2/2 RE,2/2 WA,2/2	2 4 3

Note

In sample case 3, you can find that the **runtime error** is reported by test 4 instead of 2 originally, but it's fine as long as the verdict is the same. The fifth test case is not useful even with corresponding verdicts on every solution, because it has a higher memory and time than the final result of each solution.