A. Columns Game

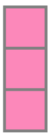
Limits: 4 sec., 1024 MiB

In this problem, you are playing an old-school video game. Columns is a match-three puzzle game. The area of play is enclosed within a tall, rectangular playfield. Columns of three colored jewels appear one at a time, at the top of the playfield and fall to the bottom, landing either on the floor or on top of previously-fallen jewels. While a column is falling, the player can move it left and right, and can also cycle the positions of the jewels within it. After a column lands, if three or more of the same color jewels are connected in a horizontal, vertical, or diagonal line, those jewels disappear. The pile of columns then settles under gravity. If this resettlement causes three or more other jewels to align, they too disappear and the cycle repeats. Occasionally, a special column containing three magic jewels appears. If it lands on a jewel, then all jewels on the playfield of the same type are cleared.

Players are scored for each match of jewels based on certain rules (please, refer to the Gameplay section below). The goal of the game is to score as much as possible before the playfield fills up with jewels, which ends the game. Go ahead and try playing the game at columns.algotester.com.

(a) Regular column

(b) Regular column



(c) Magic column

Your task is to create a game bot so that a machine can play instead of you. A game environment is defined by the sizes of the playfield and a sequence of columns your bot has to process in the specified order. Note that unlike the original game the sequence of columns is limited and known in advance. Besides, playfield sizes can be larger than the original 6 by 13 and the gameplay slightly differs from the original version.

For each column in the sequence, your bot has to decide where and how to place it. The game ends once the playfield is full or all the columns are successfully processed. Your goal is to maximize points scored by your bot.

You are given the test generator that produces a game environment based on a seed number. It is possible to check how your game bot performs on five example test cases using our game visualizer. Just click on a play icon next to one of your submissions to launch it. Note that it might take a few seconds to load the visualizer.

Input

The first line contains two integers w and h: the width and the height of the playfield, respectively. The second line contains the number of columns to process k. Each of the next k lines contains a three-character string c_i which is either `***` or consists of lowercase letters (`a-z`). Asterisks denote magic jewels while three letters denote a regular column and different letters correspond to different colors. The jewels are given in bottom-to-top order of appearance in the playfield.

Output

In the first line, print a single integer t which is the number of columns from the input sequence your bot processes. Each of the next t lines must contain two space-separated integers: p_i and s_i. Here p_i is a position of the i-th column placement and s_i is the number of cyclic shifts applied to it. Positions are numbered left-to-right from 1 to w. One cyclic shift moves the bottom-most jewel atop the column. Note that it makes no sense to apply more than two shifts to a column.

Constraints

$6 \le w \le 12$,
$13 \le h \le 19$,
$1 \le k \le 10^4$,
c_i will be either `***` or will consist of three lowercase letters (`a-z`),
$0 \le t \le k$,
$1 \le p_i \le w$,
$0 \le s_i \le 2$.

Samples

Input (*stdin*)	Output (*stdout*)
6 13 11 aba bba bbb aab bac cac *** ccb cda aad dbc	11 1 0 2 0 3 0 3 0 4 0 5 0 6 0 6 0 1 0 2 0 3 0

Notes

The sample output provided above corresponds to the quickstart solution. Please, refer to the Quickstart section for more details on how it plays the game.

Gameplay

The area of play is enclosed within a tall, rectangular w by h playfield, which is initially empty. Columns appear and are processed one by one in the given order. They are falling from the top of the playfield. A position of a column landing is determined by a player. Additionally, a player can cycle column jewels several times.

When a column lands, if its topmost jewel is outside of the play area the game ends immediately disregarding whether the column is regular or magic. Otherwise, if the current column consists of magic jewels the following happens:

1. If the magic column is landed on a jewel, then all jewels of its color will be cleared and a player will be awarded points equal to 47 $\times$ (multiplied by) the number of jewels cleared.
2. The magic jewels are cleared from the playfield.
3. All jewels above the cleared space are settled down.

Then the jewel matching starts. A *match* consists of exactly three consecutive jewels of the same color lined up either horizontally, vertically, or diagonally. Note that a single jewel can be a part of multiple matches. While there is at least one match on the playfield, all match-related jewels will be cleared. As jewels disappear from the playfield, all jewels directly above them automatically fall, potentially

causing chains of matches that help get more points. Formally, we start with a *combo multiplier* equal to 1 and while there is at least one match the following steps are repeated:

1. A player is awarded points equal to 47 × *combo multiplier* × the number of matches present.
2. Every jewel which is a part of at least one match is cleared from the playfield.
3. All jewels above the cleared space are settled down.
4. The *combo multiplier* is increased by 1.

Finally, we proceed to the next column in the sequence. If the end of the sequence has been reached, the game ends.

Scoring

If your output for a test is invalid, e.g. a column position is out of bounds, your score for the test is 0. Otherwise, your score for the test is the total number of points awarded during the game. Note that if after processing t jewels the game is still not over, every next jewel will be placed in the leftmost column with no shift applied, i.e. $p_i = 1$ and $s_i = 0$.

Your overall score is the sum of your scores over all test cases.

Submissions

- The execution time limit is 4 seconds per test case and the memory limit is 1024 mebibytes.
- The code size limit is 64 kibibytes.
- The compilation time limit is 1 minute.
- There are 50 provisional test cases. Your submissions will be evaluated on the provisional set during the submission phase.
- You can submit your code once per 10 minutes and you will get feedback with your score for each of the provisional tests.
- There will be 500 test cases in the final testing after the submission phase is over. The final results will be announced within two weeks.

Quickstart

Check the sample solution which always looks for a position where a column lands as close as possible to the bottom of the playfield. If there are multiple such positions, the leftmost will be chosen. Note that the solution implements the gameplay described above, including calculations of game points. The source code is available for some of the contest programming languages:

- C++
- Python
- Java
- C#

Tests

All test cases including provisional and final sets are generated by the generator. A test case is produced by providing the generator with a seed number. Seed number 1 corresponds to the sample case. You do NOT know seed numbers for provisional or final test sets, but you can use the generator for local testing by taking a few easy steps:

1. Check the generator source code
2. Save it to a file named generator.java
3. Compile the source code with Java

   ```
   javac generator.java
   ```

4. Generate a test case for some seed number

   ```
   java generator <seed>
   ```

   This will print the test case for `<seed>` to *standard output*.

Visualizer

For your submission, the problem visualizer can show all game states in a step-by-step mode for five example tests. To launch the visualizer, click a play icon next to your submission. Note that only tests with valid solution output are available for visualization.

The visualizer has a test selector and several settings such as play speed, markings visibility, etc. Besides, there is detailed information on the current column including its placement position and number of shifts. You can use the progress bar to scroll the game to the desired moment. Yellow marks on it correspond to columns with magic jewels in the input sequence.

Additionally, there is a manual play mode. Just visit columns.algotester.com and enjoy the game.