Problem A. **Seven-segment Display**

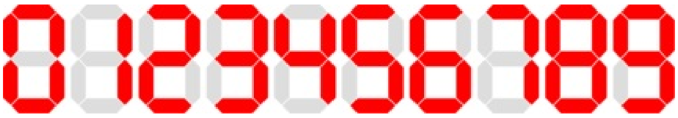
This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 8 points	Solve A-small
Large input 14 points	Solve A-large

Problem

Tom is a boy whose dream is to become a scientist, he invented a lot in his spare time. He came up with a great idea several days ago: to make a stopwatch by himself! So he bought a seven-segment display immediately.

The seven elements of the display are all light-emitting diodes (LEDs) and can be lit in different combinations to represent the arabic numerals like:

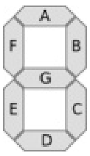


However, just when he finished the programs and tried to test the stopwatch, some of the LEDs turned out to be broken! Some of the segments can never be lit while others worked fine. So the display kept on producing some ambiguous states all the time...

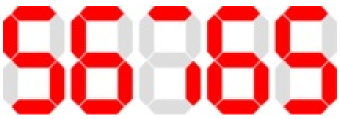
Tom has recorded a continuous sequence of states which were produced by the display and is curious about whether it is possible to understand what this display was doing. He thinks the first step is to determine the state which the display will show **next**, could you help him?

Please note that the display works well despite those broken segments, which means that the display will keep on counting down **cyclically** starting from a certain number (can be any one of 0-9 since we don't know where this record starts from). 'Cyclically' here means that each time when the display reaches 0, it will keep on counting down starting from 9 again.

For convenience, we refer the seven segments of the display by the letters A to G as the picture below:



For example, if the record of states is like:



It's not that hard to figure out that ONLY segment B is broken and the sequence of states the display is trying to produce is simply "9 -> 8 -> 7 -> 6 -> 5". Then the next number should be 4, but considering of the brokenness of segment B, the next state should be:

Input

The first line of the input gives the number of test cases, **T**. Each test case is a line containing an integer **N** which is the number of states Tom recorded and a list of the **N** states separated by spaces. Each state is encoded into a 7-character string represent the display of segment A-G, from the left to the right. Characters in the string can either be '1' or '0', denoting the corresponding segment is on or off, respectively.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1). If the input unambiguously determines the next state of the display, y should be that next state (in the same format as the input). Otherwise, y should be "ERROR!".

Limits

1 ≤ **T** ≤ 2000.

Small dataset

1 ≤ **N** ≤ 5.

Large dataset

1 ≤ **N** ≤ 100.

Sample

Input

```
4
1 1111111
2 0000000 0001010
3 0100000 0000111 0000011
5 1011011 1011111 1010000 1011111 1011011
```

Output

```
Case #1: 1110000
Case #2: ERROR!
Case #3: 0100011
Case #4: 0010011
```

Round A APAC Test

A. Seven-segment Display
B. Super 2048
C. Addition
D. Cut Tiles

Questions asked

Submissions

Seven-segment Display
8pt Not attempted **159/2058 users** correct (8%)
14pt Not attempted **34/155 users** correct (22%)

Super 2048
6pt Not attempted **875/2084 users** correct (42%)
13pt Not attempted **667/858 users** correct (78%)

Addition
11pt Not attempted **29/689 users** correct (4%)
19pt Not attempted **11/26 users** correct (42%)

Cut Tiles
13pt Not attempted **30/576 users** correct (5%)
16pt Not attempted **22/29 users** correct (76%)

Top Scores

Prowindy	100
MRain	86
Dumbear2	86
Hao.Wu	84
Gyosh	71
LinKin	71
divanshu	70
Krooonal	70
dizem	59
LMH	57

Problem B. **Super 2048**

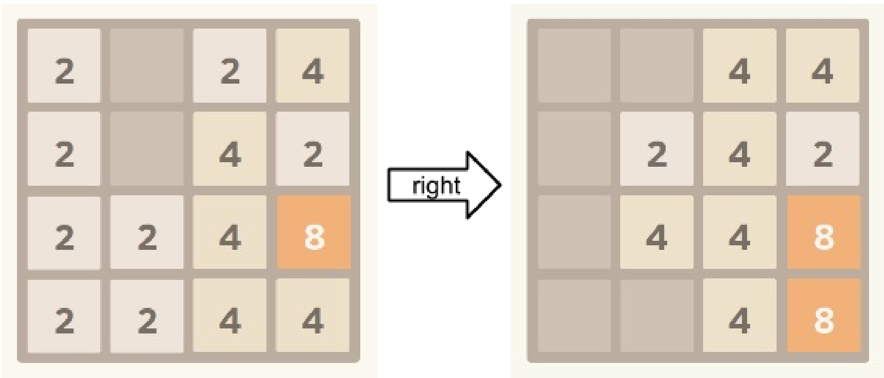
This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 6 points	Solve B-small
Large input 13 points	Solve B-large

Problem

2048 is a famous single-player game in which the objective is to slide tiles on a grid to combine them and create a tile with the number 2048.

2048 is played on a simple 4 x 4 grid with tiles that slide smoothly when a player moves them. For each movement, the player can choose to move all tiles in 4 directions, left, right, up, and down, as far as possible at the same time. If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided. **In one movement, one newly created tile can not be merged again and always is merged with the tile next to it along the moving direction first.** E.g. if the three "2" are in a row "2 2 2" and the player choose to move left, it will become "4 2 0", the most left 2 "2" are merged.



The above figure shows how 4 x 4 grid varies when player moves all tiles 'right'.

Alice and Bob accidentally find this game and love the feel when two tiles are merged. After a few round, they start to be bored about the size of the board and decide to extend the size of board to **N** x **N**, which they called the game "Super 2048".

The big board then makes them dazzled (no zuo no die -_-|). They ask you to write a program to help them figure out what the board will be looked like after all tiles move to one specific direction on a given board.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. The first line of each test case gives the side length of the board, **N**, and the direction the tiles will move to, **DIR**. **N** and **DIR** are separated by a single space. **DIR** will be one of four strings: "left", "right", "up", or "down".

The next **N** lines each contain **N** space-separated integers describing the original state of the board. Each line represents a row of the board (from top to bottom); each integer represents the value of a tile (or 0 if there is no number at that position).

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). Then output **N** more lines, each containing **N** space-separated integers which describe the board after the move in the same format as the input.

Limits

Each number in the grid is either 0 or a power of two between 2 and 1024, inclusive.

Small dataset

1 ≤ **T** ≤ 20
1 ≤ **N** ≤ 4

Large dataset

$1 \leq \mathbf{T} \leq 100$
$1 \leq \mathbf{N} \leq 20$

Sample

Input	Output
<pre>3 4 right 2 0 2 4 2 0 4 2 2 2 4 8 2 2 4 4 10 up 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 3 right 2 2 2 4 4 4 8 8 8</pre>	<pre>Case #1: 0 0 4 4 0 2 4 2 0 4 4 8 0 0 4 8 Case #2: 4 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Case #3: 0 2 4 0 4 8 0 8 16</pre>

Problem C. Addition

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 11 points — Solve C-small

Large input 19 points — Solve C-large

Problem

Six years ago, a robot, Bob, with infant's intelligence has been invented by an evil scientist, Alice.

Now the robot is six years old and studies in primary school. Addition is the first operation he learned in math. Due to his strong reasoning ability, he could now conclude a+b=12 from a=2 and b=10.

Alice wanted to test Bob's addition skills. Some equations were given to Bob in form of a=2, b=10, c=4, and Bob has to find out the answers of questions like a+b, a+c, etc.

Alice checked Bob's answers one by one in the test papers, and no mistake has been found so far, but Alice lost the given equations after a cup of coffee poured on them. However she has some of Bob's correct answers, e.g. a+b=12, a+c=6, c+d=5. She wants to continue with the checkable equations, e.g. b+d=11 could be concluded by a+b=12, a+c=6, c+d=5, and thus the question b+d is checkable.

To prevent the artificial intelligence technology from being under the control of Alice, you disguised yourself as her assistant. Now Alice wants you to figure out which of the rest of questions are checkable and their answers.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.

The first line of each test case contains a single integer **N**: the number of correctly answered questions. Each of the next **N** lines contain one correctly answered question in the form "**x**+**y**=**z**", where **x** and **y** are names of variables and **z** is a decimal integer.

The next line contains a single integer **Q**: the number of remaining questions. Each of the next **Q** lines contain one question in the form "**x**+**y**", where **x** and **y** are names of variables.

Output

For each test case, the first line of output contains "Case #**x**:", where **x** is the test case number (starting from 1). For each question in the input that was checkable, output a single line with the answer in the form "**x**+**y**=**z**", where **x** and **y** are names of variables and **z** is a decimal integer. Questions should be listed in the same order as they were given in the input. Please do **NOT** ignore duplicated questions, since Alice would fire you if you pointed any mistake of hers.

Limits

Names of variables are strings of lowercase English letters. Each name contains at most 10 characters.

-200000 ≤ **z** ≤ 200000

There is no contradiction in the answered questions and if the answer is checkable, the result is an integer.

Small dataset

T ≤ 10

N ≤ 10

Q ≤ 10

Large dataset

T ≤ 3

N ≤ 5000

Q ≤ 5000

Sample

Input	Output
`` 2 2 apple+banana=10 coconut+coconut=12 5 apple+banana apple+banana apple+apple banana+apple peach+apple 3 a+b=3 b+c=3 c+d=3 4 a+c a+d b+c b+d ```	``` Case #1: apple+banana=10 apple+banana=10 banana+apple=10 Case #2: a+d=3 b+c=3 ```

Problem D. Cut Tiles

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 13 points	Solve D-small
Large input 16 points	Solve D-large

Problem

Enzo is doing renovation for his new house. The most difficult part is to buy exactly the right number of tiles. He wants **N** tiles of different sizes. Of course they have to be cut from the tiles he bought. All the required tiles are square. The lengths of side of the tiles are $\mathbf{2^{S_1}}$, $\mathbf{2^{S_2}}$, ..., $\mathbf{2^{S_N}}$. He can only buy a lot of tiles sized **M*****M**, and he decides to only cut tiles parallel to their sides for convenience. How many tiles does he need to buy?

Input

The first line of the input gives the number of test cases: **T**. **T** lines follow. Each line start with the number **N** and **M**, indicating the number of required tiles and the size of the big tiles Enzo can buy. **N** numbers follow: $\mathbf{S_1}$, $\mathbf{S_2}$, ... $\mathbf{S_N}$, showing the sizes of the required tiles.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the number of the big tiles Enzo need to buy.

Limits

$1 \le \mathbf{2^{S_k}} \le \mathbf{M} \le 2\text{^}31-1$.

Small dataset

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{N} \le 20$.

Large dataset

$1 \le \mathbf{T} \le 1000$.
$1 \le \mathbf{N} \le 500$.

Sample

Input	Output
4 1 6 2 2 6 2 2 3 6 2 1 1 7 277 3 8 2 6 1 3 6	Case #1: 1 Case #2: 2 Case #3: 1 Case #4: 2

Top Scores

Kriiii	100
flashmt	100
adurysk	100
pulkitg10	100
cxlove321	100
Prowindy	100
ariselpy	100
Sakib	100
atony	100
kellynq	100

Problem A. Password Attacker

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 8 points — Solve A-small

Large input 13 points — Solve A-large

Problem

Passwords are widely used in our lives: for ATMs, online forum logins, mobile device unlock and door access. Everyone cares about password security. However, attackers always find ways to steal our passwords. Here is one possible situation:

Assume that Eve, the attacker, wants to steal a password from the victim Alice. Eve cleans up the keyboard beforehand. After Alice types the password and leaves, Eve collects the fingerprints on the keyboard. Now she knows which keys are used in the password. However, Eve won't know how many times each key has been pressed or the order of the keystroke sequence.

To simplify the problem, let's assume that Eve finds Alice's fingerprints only occurs on **M** keys. And she knows, by another method, that Alice's password contains **N** characters. Furthermore, every keystroke on the keyboard only generates a single, unique character. Also, Alice won't press other irrelevant keys like 'left', 'home', 'backspace' and etc.

Here's an example. Assume that Eve finds Alice's fingerprints on **M**=3 key '3', '7' and '5', and she knows that Alice's password is **N**=4-digit in length. So all the following passwords are possible: 3577, 3557, 7353 and 5735. (And, in fact, there are 32 more possible passwords.)

However, these passwords are not possible:

```
1357  // There is no fingerprint on key '1'
3355  // There is fingerprint on key '7',
         so '7' must occur at least once.
357   // Eve knows the password must be a 4-digit number.
```

With the information, please count that how many possible passwords satisfy the statements above. Since the result could be large, please output the answer modulo 1000000007(10^9+7).

Input

The first line of the input gives the number of test cases, **T**.
For the next **T** lines, each contains two space-separated numbers **M** and **N**, indicating a test case.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the total number of possible passwords modulo 1000000007(10^9+7).

Limits

Small dataset

T = 15.
1 ≤ **M** ≤ **N** ≤ 7.

Large dataset

T = 100.
1 ≤ **M** ≤ **N** ≤ 100.

Sample

```
Input      Output

4          Case #1: 1
1 1        Case #2: 36
3 4        Case #3: 120
5 5        Case #4: 674358851
15 15
```

Submissions

Password Attacker

8pt Not attempted **736/1999 users** correct (37%)

13pt Not attempted **352/627 users** correct (56%)

New Years Eve

11pt Not attempted **142/438 users** correct (32%)

12pt Not attempted **116/138 users** correct (84%)

Card Game

9pt Not attempted **750/1147 users** correct (65%)

17pt Not attempted **70/529 users** correct (13%)

Parentheses Order

10pt Not attempted **679/996 users** correct (68%)

20pt Not attempted **59/411 users** correct (14%)

Top Scores

Kriiii	100
flashmt	100
adurysk	100
pulkitg10	100
cxlove321	100
Prowindy	100
ariselpy	100
Sakib	100
atony	100
kellynq	100

Problem B. New Years Eve

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 11 points	Solve B-small
Large input 12 points	Solve B-large

Problem

At new years party there is a pyramidal arrangement of glasses for wine. For example, at the top level, there would just be one glass, at the second level there would be three, then 6 and then 10 and so on and so forth like the following image

.

The glasses are numbered using 2 numbers, **L** and **N**. **L** represents the level of the glass and **N** represents the number in that level. Numbers in a given level are as follows:

```
Level 1:
    1

Level 2:
    1
 2     3

Level 3:
      1
   2     3
4     5     6

Level 4:
         1
      2     3
   4     5     6
7     8     9     10
```

Each glass can hold **250ml** of wine. The bartender comes and starts pouring wine in the top glass(The glass numbered **L** = 1 and **N** = 1) from bottles each of capacity **750ml**.

As wine is poured in the glasses, once a glass gets full, it overflows equally into the 3 glasses on the next level below it and touching it, without any wine being spilled outside. It doesn't overflow to the glasses on the same level beside it. It also doesn't overflow to the any level below next level (directly).

For example: When the glass of **L** = 2 and **N** = 2 overflows, the water will overflow to glasses of **L** = 3 and **N** = 2, 4, 5.

Once that the bartender is done pouring **B** bottles, figure out how much quantity in ml of wine is present in the glass on level **L** with glass number **N**.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of three integers, **B**, **L**, **N**, **B** is the number of bottles the bartender pours and **L** is the glass level in the pyramid and **N** is the number of the glass in that level.

Output

For each test case, output one line containing "Case #x: y", where **x** is the test case number (starting from 1) and **y** is the quantity of wine in ml in that glass.

We recommend outputting y to 7 decimal places, but it is not required. y will be considered correct if it is close enough to the correct number: within an absolute or relative error of 10^{-6}. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

1 ≤ **T** ≤ 150.

Small dataset

1 ≤ **B** ≤ 1000.
1 ≤ **L** ≤ 100.
1 ≤ **N** ≤ Number of glasses on the corresponding level.

Large dataset

1 ≤ **B** ≤ 50000.
1 ≤ **L** ≤ 400.
1 ≤ **N** ≤ Number of glasses on the corresponding level.

Sample

Input	Output
7 1 2 1 1 1 1 2 1 1 20 1 1 1 3 1 2 3 1 10 4 10	Case #1: 166.6666667 Case #2: 250.0000000 Case #3: 250.0000000 Case #4: 250.0000000 Case #5: 0.0000000 Case #6: 55.5555556 Case #7: 157.4074074

Top Scores

Kriiii	100
flashmt	100
adurysk	100
pulkitg10	100
cxlove321	100
Prowindy	100
ariselpy	100
Sakib	100
atony	100
kellynq	100

Problem C. Card Game

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 9 points — Solve C-small

Large input 17 points — Solve C-large

Problem

Bob is fond of playing cards. On his birthday party, his best friend Alice gave him a set of cards.

There are **N** cards and each card contains an integer number. He put the cards from left to right on a desk and wants to discard some of them. Before he discards any cards, he will choose a number **K**. At each time, he always chooses 3 **adjacent** cards to discard, and we assume that the numbers on each card from left to right are **a**, **b** and **c**. Bob guarantees that

```
c - b = b - a = K
```

Bob want to know what is the smallest number of cards he can be left with at the end. If he ever has a choice of which cards to discard, he chooses the cards and will leave the fewest cards at the end.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.

Each test cases contains two lines. The first line of each test case contains two integers: the number of cards **N** and the number **K** Bob chooses. The second line contains **N** integers $\mathbf{a_1}$, $\mathbf{a_2}$, ..., $\mathbf{a_N}$ the numbers on the cards from left to right.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the smallest number of cards Bob can be left with after he has discarded everything he can.

Limits

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{a_i} \le 10^6 (1 \le i \le \mathbf{N})$.
$1 \le \mathbf{N} \le 100$.

Small dataset

$\mathbf{K} = 0$.

Large dataset

$1 \le \mathbf{K} \le 10^6$.

Sample

Input	Output
2 6 0 4 4 3 3 3 4 5 1 3 1 2 3 4	Case #1: 0 Case #2: 2

Submissions

Password Attacker

8pt	Not attempted **736/1999 users** correct (37%)
13pt	Not attempted **352/627 users** correct (56%)

New Years Eve

11pt	Not attempted **142/438 users** correct (32%)
12pt	Not attempted **116/138 users** correct (84%)

Card Game

9pt	Not attempted **750/1147 users** correct (65%)
17pt	Not attempted **70/529 users** correct (13%)

Parentheses Order

10pt	Not attempted **679/996 users** correct (68%)
20pt	Not attempted **59/411 users** correct (14%)

Top Scores

Kriiii	100
flashmt	100
adurysk	100
pulkitg10	100
cxlove321	100
Prowindy	100
ariselpy	100
Sakib	100
atony	100
kellynq	100

Problem D. Parentheses Order

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 10 points	Solve D-small
Large input 20 points	Solve D-large

Problem

An **n** parentheses sequence consists of **n** "("s and **n** ")"s.

A valid parentheses sequence is defined as the following:

You can find a way to repeat erasing adjacent pair of parentheses "()" until it becomes empty.

For example, "(())" is a valid parentheses, you can erase the pair on the 2nd and 3rd position and it becomes "()", then you can make it empty.
")()(" is not a valid parentheses, after you erase the pair on the 2nd and 3rd position, it becomes ")(" and you cannot erase any more.

Now, we have all valid **n** parentheses sequences. Find the **k**-th smallest sequence in lexicographical order.

For example, here are all valid 3 parentheses sequences in lexicographical order:

```
((()))
(()())
(())()
()(())
()()()
```

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each line represents a test case consisting of 2 integers, **n** and **k**.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the **k**-th smallest parentheses sequence in all valid **n** parentheses sequences. Output "Doesn't Exist!" when there are less than **k** different **n** parentheses sequences.

Limits

1 ≤ **T** ≤ 100.

Small dataset

1 ≤ **n** ≤ 10.
1 ≤ **k** ≤ 100000.

Large dataset

1 ≤ **n** ≤ 100.
1 ≤ **k** ≤ 10^{18}.

Sample

Input	Output
3 2 2 3 4 3 6	Case #1: ()() Case #2: ()(()) Case #3: Doesn't Exist!

Problem A. Minesweeper

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 8 points	Solve A-small
Large input 14 points	Solve A-large

Problem

Minesweeper is a computer game that became popular in the 1980s, and is still included in some versions of the Microsoft Windows operating system. This problem has a similar idea, but it does not assume you have played Minesweeper.

In this problem, you are playing a game on a grid of identical cells. The content of each cell is initially hidden. There are M mines hidden in M different cells of the grid. No other cells contain mines. You may click on any cell to reveal it. If the revealed cell contains a mine, then the game is over, and you lose. Otherwise, the revealed cell will contain a digit between 0 and 8, inclusive, which corresponds to the number of neighboring cells that contain mines. Two cells are neighbors if they share a corner or an edge. Additionally, if the revealed cell contains a 0, then all of the neighbors of the revealed cell are automatically revealed as well, recursively. When all the cells that don't contain mines have been revealed, the game ends, and you win.

For example, an initial configuration of the board may look like this ('*' denotes a mine, and 'c' is the first clicked cell):

```
*..*...**.
....*.....
..c..*....
........*.
..........
```

There are no mines adjacent to the clicked cell, so when it is revealed, it becomes a 0, and its 8 adjacent cells are revealed as well. This process continues, resulting in the following board:

```
*..*...**.
1112*.....
00012*....
00001111*.
00000001..
```

At this point, there are still un-revealed cells that do not contain mines (denoted by '.' characters), so the player has to click again in order to continue the game.

You want to win the game as quickly as possible. You want to find the minimum number of clicks to win the game. Given the size of the board (N x N), output such minimum number of clicks.

Input

The first line of the input gives the number of test cases, **T**. **T**test cases follow. First line of each test case contains one integer N. N lines strings with length N follows containing '*' and '.', denotes the Minesweeper initial board.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the minimum number of clicks to win.

Limits

1 ≤ **T** ≤ 100.

Small dataset

1 ≤ **N** ≤ 50.

Large dataset

1 ≤ **N** ≤ 300.

Sample

Input	Output

```
2       Case #1: 2
3       Case #2: 8
..*
..*
**.
5
..*..
..*..
.*..*
.*...
.*...
```

Round C APAC Test

A. Minesweeper
B. Taking Metro
C. Broken Calculator
D. Tetris

Questions asked 1

Submissions

Minesweeper
8pt Not attempted **748/1510 users** correct (50%)
14pt Not attempted **703/741 users** correct (95%)

Taking Metro
9pt Not attempted **64/319 users** correct (20%)
15pt Not attempted **56/62 users** correct (90%)

Broken Calculator
10pt Not attempted **570/1000 users** correct (57%)
16pt Not attempted **385/544 users** correct (71%)

Tetris
11pt Not attempted **29/163 users** correct (18%)
17pt Not attempted **27/29 users** correct (93%)

Top Scores

cebrusfs	100
LXZ	100
Kriiii	100
drazil	100
xing89qs	100
xhae	100
culaucon	100
whsb	100
jki14	100
zck921031	100

Problem B. **Taking Metro**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 9 points	Solve B-small
Large input 15 points	Solve B-large

Problem

Tom is taking metros in the city to go from station to station.

The metro system in the city works like this:

- There are **N** metro lines in the city: line 1, line 2, ..., line **N**.
- For each metro **i**, there are $\mathbf{SN_i}$ stations. Let's assume they are $\mathbf{S_{i,1}}$,$\mathbf{S_{i,2}}$, ... , $\mathbf{S_{i,SN_i}}$. These stations are ordered from one end point to the other end point. The metro is running in both directions. In other words, the metro is going from $\mathbf{S_{i,1}}$ -> $\mathbf{S_{i,2}}$ -> ... -> $\mathbf{S_{i,SN_i}}$, and $\mathbf{S_{i,SN_i}}$ -> $\mathbf{S_{i,SN_i-1}}$ -> ... -> $\mathbf{S_{i,1}}$. You can take the metro from any station and get off at any station. It takes a certain time to travel from one station to the next station. It takes $\mathbf{Time_{i,1}}$ minutes to travel from $\mathbf{S_{i,1}}$ to $\mathbf{S_{i,2}}$, $\mathbf{Time_{i,2}}$ minutes to travel from $\mathbf{S_{i,2}}$ to $\mathbf{S_{i,3}}$, etc. It takes the same time in the other direction.
- There are **M** transfer tunnels. Each transfer tunnel connects two stations of different metro lines. It takes a certain amount of time to travel through a tunnel in either direction. You can get off the metro at one end of the tunnel and walk through the tunnel to the station at the another end.
- When you arrive at a metro station of line **i**, you need to wait $\mathbf{W_i}$ minutes for the next metro.

Now, you are going to travel from one station to another. Find out the shortest time you need.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.

Each test case starts with an integer **N**, the number of metro lines. **N** metros descriptions follow. Each metro description starts with two integers $\mathbf{SN_i}$ and $\mathbf{W_i}$, the number of stations and the expected waiting time in minutes. The next line consists of $\mathbf{SN_i}$**-1** integers, $\mathbf{Time_{i,1}}$, $\mathbf{Time_{i,2}}$, ..., $\mathbf{Time_{i,SN_i-1}}$, describing the travel time between stations.

After the metro descriptions, there is an integer **M**, the number of tunnels. **M** lines follow to describe the tunnels. Each tunnel description consists of 5 integers, $\mathbf{m1_i}$, $\mathbf{s1_i}$, $\mathbf{m2_i}$, $\mathbf{s2_i}$, $\mathbf{t_i}$ which means the tunnel is connecting stations $\mathbf{S_{m1_i,s1_i}}$ and station $\mathbf{S_{m2_i,s2_i}}$. The walking time of the tunnel is $\mathbf{t_i}$.

The next line contains an integer **Q**, the number of queries. Each of the next **Q** lines consists of 4 integers, **x1**, **y1**, **x2**, **y2**, which mean you are going to travel from station $\mathbf{S_{x1,y1}}$ to station $\mathbf{S_{x2,y2}}$.

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1), then followed by **Q** lines, each line containing an integer y which is the shortest time you need for that query. If it's impossible, output -1 for that query instead.

Limits

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{W_i} \le 100$.
$1 \le \mathbf{Time_{i,j}} \le 100$.
$1 \le \mathbf{m1_i} \le \mathbf{N}$.
$1 \le \mathbf{s1_i} \le \mathbf{SN_{m1_i}}$.
$1 \le \mathbf{m2_i} \le \mathbf{N}$.
$1 \le \mathbf{s2_i} \le \mathbf{SN_{m2_i}}$.
$\mathbf{m1_i}$ and $\mathbf{m2_i}$ will be different.
$1 \le \mathbf{t_i} \le 100$.
$1 \le \mathbf{Q} \le 10$.
$1 \le \mathbf{x1} \le \mathbf{N}$.
$1 \le \mathbf{y1} \le \mathbf{SN_{x1}}$.

$1 \le \mathbf{x2} \le \mathbf{N}$.
$1 \le \mathbf{y2} \le \mathbf{SN_{y2}}$.
Station $\mathbf{S_{x1,y1}}$ and station $\mathbf{S_{x2,y2}}$ will be different.

Small dataset

$1 \le \mathbf{N} \le 10$.
$0 \le \mathbf{M} \le 10$.
$2 \le \mathbf{SN_i} \le 100$.
The total number of stations in each case is at most 100.

Large dataset

$1 \le \mathbf{N} \le 100$.
$0 \le \mathbf{M} \le 100$.
$2 \le \mathbf{SN_i} \le 1000$.
The total number of stations in each case is at most 1000.

Sample

Input

```
2

2
5 3
3 5 7 3
4 2
1 1 1
1
1 2 2 2 1
1
1 1 2 4

2
5 3
3 5 7 3
4 2
1 1 1
2
1 2 2 2 1
2 4 1 4 1
1
1 1 1 5
```

Output

```
Case #1:
11
Case #2:
18
```

In the first case, you are going to travel from station 1 of metro line 1 to station 4 of metro line 2. The best way is:

- wait 3 minutes for line 1 and get on it.
- take it for 3 minutes and get off at station 2.
- take the tunnel and walk for 1 minute to station 2 of line 2.
- wait 2 minutes for line 2 and get on it.
- take it for 2 minutes and get off at station 4.

The total time is: 3+3+1+2+2=11.

Problem C. Broken Calculator

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input
10 points
Solve C-small

Large input
16 points
Solve C-large

Problem

Alice is a smart student who is very good at math. She is attending a math class. In this class, the teacher is teaching the students how to use a calculator. The teacher will tell an integer to all of the students, and the students must type that exact number into their calculators. If someone fails to type the number, he or she will be punished for failing such an easy task!

Unfortunately, at the beginning of the class, Alice finds that her calculator is broken! She finds that some of the number buttons are totally broken, and only the "multiply" and "equals" operator buttons are available to use. So she can only use these buttons to get the number quickly.

For instance, the teacher may say the number "60", while Alice's calculator can only type "1", "2" and "5". She could push the following buttons:

- Button "15" (2 clicks)
- Button "multiply" (1 click)
- Button "2" (1 click)
- Button "multiply" (1 click)
- Button "2" (1 click)
- Button "equals" (1 click)

This method requires 7 button clicks. However, if Alice uses "12*5=", only 5 clicks are needed. Of course Alice wants to get the integer as fast as possbile, so she wants to minimize the number of button clicks. Your task is to help her find a way to get the required number quickly.

Input

The first line of the input gives a number **T**, the number of integers the teacher says. **T** test cases follow.

Each case contains two lines. The first line contains ten numbers each of which is only 0 or 1. the **i**th number (starting from 0) is "1" if the number **i** can be clicked, or "0" if it is broken. The second line contains only one number **X**, the integer the teacher tells everyone.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the minimum number of button clicks needed, or "Impossible" if it is not possible to produce the number.

Limits

$1 \le \mathbf{T} \le 100$.

Small dataset

$1 \le \mathbf{X} \le 100$.

Large dataset

$1 \le \mathbf{X} \le 10^6$.

Sample

Input

```
3
0 1 1 0 0 1 0 0 0 0
60
1 1 1 1 1 1 1 1 1 1
128
0 1 0 1 0 1 0 1 0 1
128
```

Output

```
Case #1: 5
Case #2: 4
Case #3: Impossible
```

The first sample case is explained in problem statement.

In the second case, all digits are available, so Alice can just press "1", "2", "8" and then "equals" to get the result. Please note that she still needs to press "equals" in the last step, even though there are no calculations.

For the last case, it's impossible since Alice cannot input any even numbers.

Round C APAC Test

A. Minesweeper

B. Taking Metro

C. Broken Calculator

D. Tetris

Questions asked 1

Submissions

Minesweeper

8pt Not attempted **748/1510 users** correct (50%)

14pt Not attempted **703/741 users** correct (95%)

Taking Metro

9pt Not attempted **64/319 users** correct (20%)

15pt Not attempted **56/62 users** correct (90%)

Broken Calculator

10pt Not attempted **570/1000 users** correct (57%)

16pt Not attempted **385/544 users** correct (71%)

Tetris

11pt Not attempted **29/163 users** correct (18%)

17pt Not attempted **27/29 users** correct (93%)

Top Scores

cebrusfs	100
LXZ	100
Kriiii	100
drazil	100
xing89qs	100
xhae	100
culaucon	100
whsb	100
jki14	100
zck921031	100

Problem D. **Tetris**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input
11 points

Solve D-small

Large input
17 points

Solve D-large

Problem

Tetris is a famous video game that almost everyone has played it. In this problem, you need to simulate a simplified version of it.

In our version, the game is played in a **W** by **H** field with gravity. At the beginning, the field is empty. Then the tetrominos start to fall from above top of the field to bottom of the field, one by one. Each tetromino will stop as soon as it touches some other tetrominos or bottom of the field.

One interesting feature of the game is called "line clearing". A line will be cleared as soon as it is filled by tetrominos. More than one line may be cleared at a time. For example:

```
|..............|      |..............|      |..............|
|.............o|      |..............|      |..............|
|.............o|      |..............|      |..............|
|.............o|      |..............|      |..............|
|.............o|      |..............|      |..............|
|..xx..........| -->  |..xx..........| -->  |..............|
|xxxxxxxxxxxxx.|      |xxxxxxxxxxxxxo|      |..............|
|xxxxxxxxxxxxx.|      |xxxxxxxxxxxxxo|      |..xx..........|
|xx..xxxxxxxxx.|      |xx..xxxxxxxxxo|      |xx..xxxxxxxxxo|
|xxxxxxxxxxx...|      |xxxxxxxxxxx..o|      |xxxxxxxxxxx..o|
----------------      ----------------      ----------------

Falling               Stopped               Cleared 2 lines
```

Note that in this simplified version, the "floating" tetromino blocks won't continue to fall after lines are cleared. This is why the top-most two squares will keep in such position. Consequently, cascade clearing won't happen, even though it would happen in the original version of Tetris.

The game ends when all the given tetrominos are placed, or the current tetromino cannot be placed due to the height limit of the field is reached.

In this problem, each tetromino will has its type, rotation and falling position told by the input. They will start to fall from the **above** of the field. Your goal is to simulate and get the final result of each play.

Input

We have 7 types of tetromino:

```
1   2   3   4   5   6   7

x    x  x    x  xx  x    x
xx  xx  x    x  xx  x   xxx
 x  x   xx  xx      x
                    x
```

Rotation of a tetromino is represented by a number **r**. **r** can be 0, 1, 2 or 3. Rotation is counterclockwise. For example:

```
r=0   r=1  r=2   r=3

  x     x   xxx   x
 xxx   xx    x    xx
        x         x

 x     xx   x     xx
 xx   xx    xx   xx
  x          x
```

The horizontal falling position is represented by a number **x**. It is the coordinate of the lower left square of a tetromino's bounding box. Here **x** starts from 0.

The first line of the input gives the number of test cases, **T**. For each test case, the first line of input has 3 integers, **W, H, N**. **W** is the width, **H** is the height and **N** is the number of blocks that are going to fall.

Then **N** lines below, each line has 3 integers, $\mathbf{t_i}$, $\mathbf{r_i}$, $\mathbf{x_i}$. $\mathbf{t_i}$ tells the tetromino type. $\mathbf{r_i}$ is the rotation of this tetromino. $\mathbf{x_i}$ is the horizontal falling position of this tetromino. It is guaranteed that $\mathbf{x_i}$ will make the tetromino inside the field, horizontally.

Output

For each test case, first output one line containing "Case #i:", where **i** is the test case number (starting from 1). And then, if the game ends before the **N** blocks, output "Game Over!"(without quotes). Otherwise, output the game field's final state, which should have **H** lines, each has **W** characters. Each character can be '.' or 'x'.

Limits

1 <= **T** <= 100
1 <= $\mathbf{t_i}$ <= 7
0 <= $\mathbf{r_i}$ < 4

Small dataset

4 <= **W** <= 20
1 <= **H** <= 20
0 <= **N** <= 100

Large dataset

4 <= **W** <= 100
1 <= **H** <= 100
0 <= **N** <= 5000

Sample

Input

```
5
8 6 1
1 0 0
5 4 1
1 1 1
5 6 3
5 0 0
5 0 2
3 2 3
6 4 3
6 2 0
6 2 0
6 2 0
6 4 2
6 0 0
6 0 1
```

Output

```
Case #1:
........
........
........
x.......
xx......
.x......
Case #2:
.....
.....
..xx.
.xx..
Case #3:
.....
.....
.....
.....
.....
...xx
Case #4:
Game Over!
Case #5:
xx....
xx....
xx....
xx....
```

Problem A. **Cube IV**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input
8 points
Solve A-small

Large input
15 points
Solve A-large

Problem

Vincenzo decides to make cube IV but only has the budget to make a square maze. Its a perfect maze, every room is in the form of a square and there are 4 doors (1 on each side of the room). There is a big number written in the room. A person can only move from one room to another if the number in the next room is larger than the number in his current room by 1. Now, Vincenzo assigns unique numbers to all the rooms (1, 2, 3, ... S^2) and then places S^2 people in the maze, 1 in each room where S is the side length of the maze. The person who can move maximum number of times will win. Figure out who will emerge as the winner and the number of rooms he will be able to move.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of **S** which is the side length of the square maze. Then S^2 numbers follow like a maze to give the numbers that have been assigned to the rooms.

```
1 2 9
5 3 8
4 6 7
```

Output

For each test case, output one line containing "Case #x: r d", where x is the test case number (starting from 1), r is the room number of the person who will win and d is the number of rooms he could move. In case there are multiple such people, the person who is in the smallest room will win.

Limits

1 ≤ **T** ≤ 100.

Small dataset

1 ≤ **S** ≤ 10

Large dataset

1 ≤ **S** ≤ 10^3.

Sample

Input	Output
<pre>2 2 3 4 1 2 3 1 2 9 5 3 8 4 6 7</pre>	<pre>Case #1: 1 2 Case #2: 6 4</pre>

Round D APAC Test

A. Cube IV
B. GBus count
C. Sort a scrambled itinerary
D. Itz Chess

Questions asked 4

Submissions

Cube IV

8pt	Not attempted **1708/2380 users** correct (72%)
15pt	Not attempted **1492/1679 users** correct (89%)

GBus count

9pt	Not attempted **2048/2354 users** correct (87%)
15pt	Not attempted **1865/2018 users** correct (92%)

Sort a scrambled itinerary

11pt	Not attempted **1623/1914 users** correct (85%)
15pt	Not attempted **1483/1602 users** correct (93%)

Itz Chess

12pt	Not attempted **654/1008 users** correct (65%)
15pt	Not attempted **393/622 users** correct (63%)

Top Scores

dreamoon	100
Kriiii	100
Balajiganapathi	100
uws933	100
NExPlain	100
culaucon	100
fahimzubayer18	100
pattara.s	100
buaamm	100
lijiancheng	100

Problem B. GBus count

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 9 points	Solve B-small
Large input 15 points	Solve B-large

Problem

There exists a straight line along which cities are built.

Each city is given a number starting from 1. So if there are 10 cities, city 1 has a number 1, city 2 has a number 2,... city 10 has a number 10.

Different buses (named GBus) operate within different cities, covering all the cities along the way. The cities covered by a GBus are represented as 'first_city_number last_city_number' So, if a GBus covers cities 1 to 10 inclusive, the cities covered by it are represented as '1 10'

We are given the cities covered by all the GBuses. We need to find out how many GBuses go through a particular city.

Input

The first line contains the number of test cases (**T**), after which **T** cases follow each separated from the next with a **blank** line.
For each test case,
The first line contains the number of GBuses.(**N**)
Second line contains the cities covered by them in the form
$\mathbf{a_1\ b_1\ a_2\ b_2\ a_3\ b_3 ... a_n\ b_n}$
where GBus1 covers cities numbered from a_1 to b_1, GBus2 covers cities numbered from a_2 to b_2, GBus3 covers cities numbered from a_3 to b_3, upto **N** GBuses.
Next line contains the number of cities for which GBus count needs to be determined (**P**).
The below **P** lines contain different city numbers.

Output

For each test case, output one line containing "Case #T_i:" followed by **P** numbers corresponding to the number of cities each of those **P** GBuses goes through.

Limits

1 <= **T** <= 10
$\mathbf{a_i}$ and $\mathbf{b_i}$ will always be integers.

Small dataset

1 <= **N** <= 50
1 <= $\mathbf{a_i}$ <= 500, 1 <= $\mathbf{b_i}$ <= 500
1 <= **P** <= 50

Large dataset

1 <= **N** <= 500
1 <= $\mathbf{a_i}$ <= 5000, 1 <= $\mathbf{b_i}$ <= 5000
1 <= **P** <= 500

Sample

Input

```
2
4
15 25 30 35 45 50 10 20
2
15
25

10
10 15 5 12 40 55 1 10 25 35 45 50 20 28 27 35 15 40 4 5
3
5

10
27
```

Output

```
Case #1: 2 1
Case #2: 3 3 4
```

Explanation for case 1:
2 GBuses go through city 15 (GBus1 [15 25] and GBus4 [10 20])
1 GBus goes through city 25 (GBus1 [15 25])

Round D APAC Test
A. Cube IV
B. GBus count
C. Sort a scrambled itinerary
D. Itz Chess

Questions asked 4

Submissions

Cube IV
8pt Not attempted 1708/2380 users correct (72%)
15pt Not attempted 1492/1679 users correct (89%)

GBus count
9pt Not attempted 2048/2354 users correct (87%)
15pt Not attempted 1865/2018 users correct (92%)

Sort a scrambled itinerary
11pt Not attempted 1623/1914 users correct (85%)
15pt Not attempted 1483/1602 users correct (93%)

Itz Chess
12pt Not attempted 654/1008 users correct (65%)
15pt Not attempted 393/622 users correct (63%)

Top Scores

dreamoon	100
Kriiii	100
Balajiganapathi	100
uws933	100
NExPlain	100
culaucon	100
fahimzubayer18	100
pattara.s	100
buaamm	100
lijiancheng	100

Problem C. Sort a scrambled itinerary

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 11 points	Solve C-small
Large input 15 points	Solve C-large

Problem

Once upon a day, Mary bought a one-way ticket from somewhere to somewhere with some flight transfers.

For example: SFO->DFW DFW->JFK JFK->MIA MIA->ORD.

Obviously, transfer flights at a city twice or more doesn't make any sense. So Mary will not do that.

Unfortunately, after she received the tickets, she messed up the tickets and she forgot the order of the ticket.

Help Mary rearrange the tickets to make the tickets in correct order.

Input

The first line contains the number of test cases **T**, after which **T** cases follow.
For each case, it starts with an integer **N**. There are **N** flight tickets follow.
Each of the next 2 lines contains the source and destination of a flight ticket.

Output

For each test case, output one line containing "Case #x: itinerary", where **x** is the test case number (starting from 1) and **itinerary** is sorted list of flight tickets which represents the actual itinerary. Each flight segment in the itinerary should be outputted as pair of source-destination airport codes.

Limits

$1 \le \mathbf{T} \le 100$.
For each case, the input tickets are messed up from an entire itinerary bought by Mary. In other words, it's ensured can be recovered to a valid itinerary.

Small dataset

$1 \le \mathbf{N} \le 100$.

Large dataset

$1 \le \mathbf{N} \le 10^4$.

(The segment for second case in sample can be seen as below) MIA-ORD, DFW-JFK, SFO-DFW, JFK-MIA

Sample

Input	Output
2 1 SFO DFW 4 MIA ORD DFW JFK SFO DFW JFK MIA	Case #1: SFO-DFW Case #2: SFO-DFW DFW-JFK JFK-MIA MIA-ORD

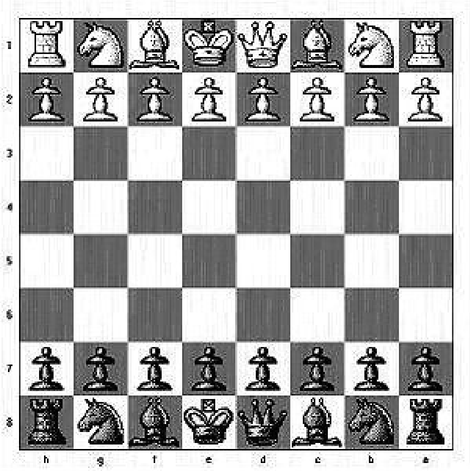
Problem D. **Itz Chess**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 12 points	Solve D-small
Large input 15 points	Solve D-large

Problem

Given an arranged chess board with pieces, figure out the total number of different ways in which any piece can be killed **in one move**. Note: in this problem, the pieces can be killed despite of the color.



For example, if there are 3 pieces King is at B2, Pawn at A1 and Queen at H8 then the total number of pieces that an be killed is 3. H8-Q can kill B2-K, A1-P can kill B2-K, B2-K can kill A1-P

A position on the chess board is represented as A1, A2... A8,B1.. H8

Pieces are represented as

- (K) King can move in 8 direction by one place.
- (Q) Queen can move in 8 direction by any number of places, but can't overtake another piece.
- (R) Rook can only move vertically or horitonzally, but can't overtake another piece.
- (B) Bishop can only move diagonally, but can't overtake another piece.
- (N) Knights can move to a square that is two squares horizontally and one square vertically **OR** one squares horizontally and two square vertically.
- (P) Pawn can only kill by moving diagonally upwards (towards higher number i.e. A -> B, B->C and so on).

Input

The first line of the input gives the number of test cases, **T**. **T** Test cases follow. Each test case consists of the number of pieces , **N**. **N** lines follow, each line mentions where a piece is present followed by **-** with the piece type

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the the total number of different ways in which any piece can be killed.

Limits

1 ≤ **T** ≤ 100.

Small dataset

1 ≤ **N** ≤ 10.
Pieces can include K, P

Large dataset

1 ≤ **N** ≤ 64.

Sample

```
Input     Output

2         Case #1: 1
2         Case #2: 3
A1-K
A8-Q

3
B2-K
A1-P
H8-Q
```