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

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

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

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

Output

```
Case #1: 0
Case #2: 2
```

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

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

Small dataset

$1 \le \mathbf{n} \le 10$.
$1 \le \mathbf{k} \le 100000$.

Large dataset

$1 \le \mathbf{n} \le 100$.
$1 \le \mathbf{k} \le 10^{18}$.

Sample

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