Problem A. Alien Language

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 A-small
Large input 23 points	Solve A-large

Problem

After years of study, scientists at Google Labs have discovered an alien language transmitted from a faraway planet. The alien language is very unique in that every word consists of exactly **L** lowercase letters. Also, there are exactly **D** words in this language.

Once the dictionary of all the words in the alien language was built, the next breakthrough was to discover that the aliens have been transmitting messages to Earth for the past decade. Unfortunately, these signals are weakened due to the distance between our two planets and some of the words may be misinterpreted. In order to help them decipher these messages, the scientists have asked you to devise an algorithm that will determine the number of possible interpretations for a given pattern.

A pattern consists of exactly **L** tokens. Each token is either a single lowercase letter (the scientists are very sure that this is the letter) or a group of unique lowercase letters surrounded by parenthesis (and). For example: (ab)d(dc) means the first letter is either a or b, the second letter is definitely d and the last letter is either d or c. Therefore, the pattern (ab)d(dc) can stand for either one of these 4 possibilities: add, adc, bdd, bdc.

Input

The first line of input contains 3 integers, **L**, **D** and **N** separated by a space. **D** lines follow, each containing one word of length **L**. These are the words that are known to exist in the alien language. **N** test cases then follow, each on its own line and each consisting of a pattern as described above. You may assume that all known words provided are unique.

Output

For each test case, output

```
Case #X: K
```

where **X** is the test case number, starting from 1, and **K** indicates how many words in the alien language match the pattern.

Limits

Small dataset

$1 \le \mathbf{L} \le 10$
$1 \le \mathbf{D} \le 25$
$1 \le \mathbf{N} \le 10$

Large dataset

$1 \le \mathbf{L} \le 15$
$1 \le \mathbf{D} \le 5000$
$1 \le \mathbf{N} \le 500$

Sample

Input	Output
3 5 4 abc bca dac dbc cba (ab)(bc)(ca) abc (abc)(abc)(abc) (zyx)bc	Case #1: 2 Case #2: 1 Case #3: 3 Case #4: 0

Problem B. **Watersheds**

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 B-small
Large input 23 points	Solve B-large

Problem

Geologists sometimes divide an area of land into different regions based on where rainfall flows down to. These regions are called *drainage basins*.

Given an elevation map (a 2-dimensional array of altitudes), label the map such that locations in the same drainage basin have the same label, subject to the following rules.

- From each cell, water flows down to at most one of its 4 neighboring cells.
- For each cell, if none of its 4 neighboring cells has a lower altitude than the current cell's, then the water does not flow, and the current cell is called a *sink*.
- Otherwise, water flows from the current cell to the neighbor with the lowest altitude.
- In case of a tie, water will choose the first direction with the lowest altitude from this list: North, West, East, South.

Every cell that drains directly or indirectly to the same sink is part of the same drainage basin. Each basin is labeled by a unique lower-case letter, in such a way that, when the rows of the map are concatenated from top to bottom, the resulting string is lexicographically smallest. (In particular, the basin of the most North-Western cell is always labeled 'a'.)

Input

The first line of the input file will contain the number of maps, **T**. **T** maps will follow, each starting with two integers on a line -- **H** and **W** -- the height and width of the map, in cells. The next **H** lines will each contain a row of the map, from north to south, each containing **W** integers, from west to east, specifying the altitudes of the cells.

Output

For each test case, output 1+**H** lines. The first line must be of the form

```
Case #X:
```

where **X** is the test case number, starting from 1. The next **H** lines must list the basin labels for each of the cells, in the same order as they appear in the input.

Limits

T ≤ 100;

Small dataset

1 ≤ **H**, **W** ≤ 10;
0 ≤ altitudes < 10.
There will be at most two basins.

Large dataset

1 ≤ **H**, **W** ≤ 100;
0 ≤ altitudes < 10,000.
There will be at most 26 basins.

Sample

Input	Output
5 3 3 9 6 3 5 9 6 3 5 9 1 10 0 1 2 3 4 5 6 7 8 7 2 3	Case #1: a b b a a b a a a Case #2: a a a a a a a a a b Case #3: a a a

```
7 6 7
7 6 7
5 5
1 2 3 4 5
2 9 3 9 6
3 3 0 8 7
4 9 8 9 8
5 6 7 8 9
2 13
8 8 8 8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8 8 8 8
```

```
b b b
Case #4:
a a a a a
a a b b a
a b b b a
a b b b a
a a a a a
Case #5:
a b c d e f g h i j k l m
n o p q r s t u v w x y z
```

Notes

In Case #1, the upper-right and lower-left corners are sinks. Water from the diagonal flows towards the lower-left because of the lower altitude (5 versus 6).

Problem C. Welcome to Code Jam

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 23 points	Solve C-large

Problem

So you've registered. We sent you a welcoming email, to welcome you to code jam. But it's possible that you still don't feel welcomed to code jam. That's why we decided to name a problem "welcome to code jam." After solving this problem, we hope that you'll feel very welcome. Very welcome, that is, to code jam.

If you read the previous paragraph, you're probably wondering why it's there. But if you read it very carefully, you might notice that we have written the words "welcome to code jam" several times: 400263727 times in total. After all, it's easy to look through the paragraph and find a 'w'; then find an 'e' later in the paragraph; then find an 'l' after that, and so on. Your task is to write a program that can take any text and print out how many times that text contains the phrase "welcome to code jam".

To be more precise, given a text string, you are to determine how many times the string "welcome to code jam" appears as a sub-sequence of that string. In other words, find a sequence *s* of increasing indices into the input string such that the concatenation of input[s[0]], input[s[1]], ..., input[s[18]] is the string "welcome to code jam".

The result of your calculation might be huge, so for convenience we would only like you to find the last 4 digits.

Input

The first line of input gives the number of test cases, **N**. The next N lines of input contain one test case each. Each test case is a single line of text, containing only lower-case letters and spaces. No line will start with a space, and no line will end with a space.

Output

For each test case, "Case #x: dddd", where x is the case number, and dddd is the last four digits of the answer. If the answer has fewer than 4 digits, please add zeroes at the front of your answer to make it exactly 4 digits long.

Limits

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

Small dataset

Each line will be no longer than 30 characters.

Large dataset

Each line will be no longer than 500 characters.

Sample

Input	Output
3 elcomew elcome to code jam wweellccoommee to code qps jam welcome to codejam	Case #1: 0001 Case #2: 0256 Case #3: 0000

Problem A. Multi-base happiness

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 A-small
Large input 18 points	Solve A-large

Problem

Given an integer N, replace it by the sum of the squares of its digits. A happy number is a number where, if you apply this process repeatedly, it eventually results in the number 1. For example, if you start with 82:

```
8*8 + 2*2       = 64 + 4    = 68,  repeat:
6*6 + 8*8       = 36 + 64   = 100, repeat:
1*1 + 0*0 + 0*0 = 1 + 0 + 0 = 1 (happy! :)
```

Since this process resulted in 1, 82 is a happy number.

Notice that a number might be happy in some bases, but not happy in others. For instance, the base 10 number 82 is not a happy number when written in base 3 (as 10001).

You are one of the world's top number detectives. Some of the bases got together (yes, they are organized!) and hired you for an important task: find out what's the smallest integer number that's greater than 1 and is happy in all the given bases.

Input

The first line of input gives the number of cases **T**. **T** test cases follow. Each case consists of a single line. Each line contains a space separated list of distinct integers, representing the bases. The list of bases is always in increasing order.

Output

For each test case, output:

```
Case #X: K
```

where **X** is the test case number, starting from 1, and **K** is the decimal representation of the smallest integer (greater than 1) which is happy in all of the given bases.

Limits

2 ≤ all possible input bases ≤ 10

Small dataset

1 ≤ **T** ≤ 42

2 ≤ number of bases on each test case ≤ 3

Large dataset

1 ≤ **T** ≤ 500

2 ≤ number of bases on each test case ≤ 9

Sample

Input	Output
3 2 3 2 3 7 9 10	Case #1: 3 Case #2: 143 Case #3: 91

Important Note

Please remember that you must submit all code used to solve the problem.

Problem B. **Crossing the Road**

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.

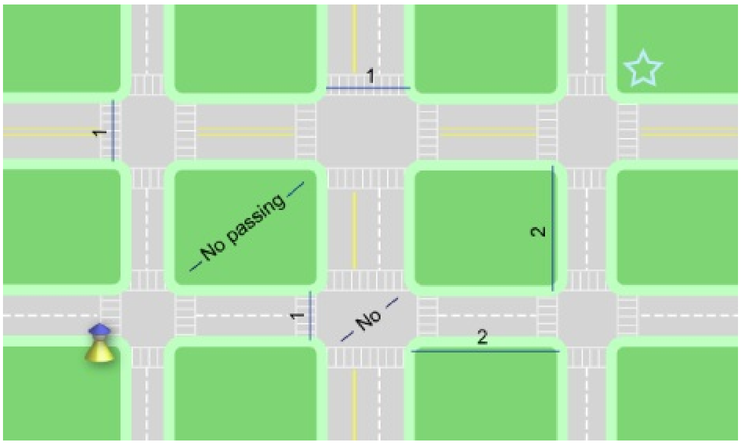
Small input 13 points	Solve B-small
Large input 20 points	Solve B-large

Problem

Where roads intersect, there are often traffic lights that tell pedestrians (people walking) when they should cross the street. A clever pedestrian may try to optimize her path through a city based on when those lights turn green.

The city in this problem is a grid, **N** rows tall by **M** columns wide. Our pedestrian wants to get from the northeast corner of the southwest block to the southwest corner of the northeast block. Your objective is to help her find her way from corner to corner in the fastest way possible.

The pedestrian can cross a street in 1 minute, but only if the traffic light is green for the entire crossing. The pedestrian can move between two streets, along one edge of a block, in 2 minutes. The pedestrian can only move along the edges of the block; she cannot move diagonally from one corner of a block to the opposite corner.



Traffic lights follow the following pattern: at intersection *i*, the north-south lights stay green for $\mathbf{S_i}$ minutes, while the east-west lights stay red. Then the north-south lights turn red, the east-west lights turn green, and they stay that way for $\mathbf{W_i}$ minutes. Then they start the same cycle again. The pedestrian starts moving at t=0 minutes; traffic light i starts a cycle by turning green in the north-south direction at t=$\mathbf{T_i}$ minutes. There are cycles before t=T_i as well.

For example, intersection 0 could have the following values:

$S_0 = 3,\ W_0 = 2,\ T_0 = 0$

The north-south direction turns green after 0 minutes. That lasts 3 minutes, during which time the pedestrian can cross in the north-south direction and not the east-west direction. Then the lights switch, and for the next 2 minutes the pedestrian can cross in the east-west direction and not the north-south direction. Then, 5 minutes after it started, the cycle starts again. This is exactly the same as the following configuration:

$S_0 = 3,\ W_0 = 2,\ T_0 = 10$

Input

The first line in the input contains the number of test cases, **C**. This is followed by **C** test cases in the following format:

A single line containing "**N M**", where **N** and **M** are the number of horizontal roads (rows) and vertical roads (columns), as above. This is followed by **N** lines. The *i*th of those lines contains information about intersections on the *i*th row, where the 0th row is the northmost. Each of those lines will contain 3**M** integers, separated by spaces, in the following form:

$S_{i,0}\ W_{i,0}\ T_{i,0}\ S_{i,1}\ W_{i,1}\ T_{i,1} \ldots\ S_{i,M-1}\ W_{i,M-1}\ T_{i,M-1}$

$\mathbf{S}_{i,j}$, $\mathbf{W}_{i,j}$ and $\mathbf{T}_{i,j}$ all refer to the intersection in the ith row from the north and the jth column from the west.

Output

For each test case, output a single line containing the text "Case #x: t", where x is the number of the test case and t is the minimum number of minutes it takes the pedestrian to get from the southwest corner to the northeast corner.

Limits

C, **N**, **M**, $\mathbf{S}_{i,j}$, $\mathbf{W}_{i,j}$, $\mathbf{T}_{i,j}$ are all non-negative integers.
$\mathbf{C} \le 100$

Small Input

$1 \le \mathbf{N}, \mathbf{M} \le 3$
$0 < \mathbf{S}_{i,j}, \mathbf{W}_{i,j} \le 10$
$0 \le \mathbf{T}_{i,j} \le 20$

Large Input

$1 \le \mathbf{N}, \mathbf{M} \le 20$
$0 < \mathbf{S}_{i,j}, \mathbf{W}_{i,j} \le 10^7$
$0 \le \mathbf{T}_{i,j} \le 10^8$

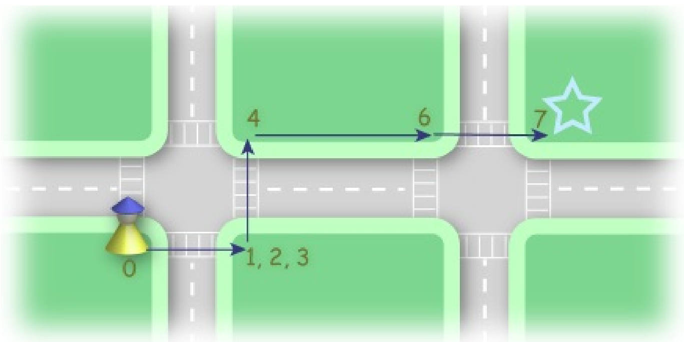
Sample

Input	Output
2 1 1 3 2 10 1 2 1 5 3 1 5 2	Case #1: 4 Case #2: 7

Explanation

The first case is described above. The pedestrian crosses to the North (1 minute), waits 2 minutes and then crosses to the East (1 minute), for a total of 4 minutes.

The second case is depicted in the diagram below. The pedestrian crosses to the East (1 minute), waits 2 minutes and crosses to the North (1 minute). Then she walks east a block (2 minutes) and crosses to the East (1 minute) for a total of 7 minutes.



Problem C. Collecting Cards

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 30 points	Solve C-large

Problem

You've become addicted to the latest craze in collectible card games, PokeCraft: The Gathering. You've mastered the rules! You've crafted balanced, offensive, and defensive decks! You argue the merits of various cards on Internet forums! You compete in tournaments! And now, as they just announced their huge new set of cards coming in the year 2010, you've decided you'd like to collect every last one of them! Fortunately, the one remaining sane part of your brain is wondering: how much will this cost?

There are **C** kinds of card in the coming set. The cards are going to be sold in "booster packs", each of which contains **N** cards of different kinds. There are many possible combinations for a booster pack where no card is repeated. When you pay for one pack, you will get any of the possible combinations with equal probability. You buy packs one by one, until you own all the **C** kinds. What is the expected (average) number of booster packs you will need to buy?

Input

The first line of input gives the number of cases, **T**. **T** test cases follow, each consisting of a line containing **C** and **N**.

Output

For each test case, output one line in the form

```
Case #x: E
```

where **x** is the case number,starting from 1, and **E** is the expected number of booster packs you will need to buy. Any answer with a relative or absolute error at most 10^{-5} will be accepted.

Limits

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

Small dataset

$1 \le \mathbf{N} \le \mathbf{C} \le 10$

Large dataset

$1 \le \mathbf{N} \le \mathbf{C} \le 40$

Sample

Input	Output
2 2 1 3 2	Case #1: 3.0000000 Case #2: 2.5000000

Submissions

Decision Tree

10pt	Not attempted **1512/1752 users** correct (86%)
11pt	Not attempted **1266/1544 users** correct (82%)

The Next Number

9pt	Not attempted **2559/3329 users** correct (77%)
26pt	Not attempted **1890/2557 users** correct (74%)

Square Math

12pt	Not attempted **157/422 users** correct (37%)
32pt	Not attempted **69/168 users** correct (41%)

Top Scores

ACRush	100
ftc	100
bmerry	100
andrewzta	100
ipknHama	100
halyavin	100
mystic	100
Yarin	100
Khuc.Anh.Tuan	100
dgozman	100

Problem A. Decision Tree

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 A-small
Large input 11 points	Solve A-large

Problem

Decision trees -- in particular, a type called classification trees -- are data structures that are used to classify *items* into *categories* using *features* of those items. For example, each animal is either "cute" or not. For any given animal, we can decide whether it is cute by looking at the animal's features and using the following decision tree.

```
(0.2 furry
  (0.81 fast
    (0.3)
    (0.2)
  )
  (0.1 fishy
    (0.3 freshwater
      (0.01)
      (0.01)
    )
    (0.1)
  )
)
```

A decision tree is defined recursively. It always has a root node and a weight. It also, optionally, has a feature name and two sub-trees, which are themselves decision trees.

More formally, a decision tree is defined using the following grammar.

```
tree ::= (weight [feature tree tree])
weight is a real number between 0 and 1, inclusive
feature is a string of 1 or more lower case English letters
```

The part inside the square brackets, **[]**, is optional. The parentheses, (), *weight* and *feature* are tokens. There will be at least one whitespace character between any two tokens, except (possibly) after an open-bracket '(' or before a close-bracket ')'. Whitespace characters are space (' ') and endline ('\n').

To figure out how likely the animal is to be cute, we start at the root of the tree with probability *p* set to 1. At each node, we multiply *p* by the weight of the node. If the node is a leaf (has no sub-trees), then we stop, and the value of *p* is the probability that our animal is cute. Otherwise, we look at the feature associated with the node. If our animal has this feature, we move down into the first sub-tree and continue recursively. If it does not have this feature, then we move down into the second sub-tree and continue in the same way.

For example, a beaver is an animal that has two features: *furry* and *freshwater*. We start at the root with *p* equal to 1. We multiply *p* by 0.2, the weight of the root and move into the first sub-tree because the beaver has the *furry* feature. There, we multiply *p* by 0.81, which makes *p* equal to 0.162. From there we move further down into the second sub-tree because the beaver does not have the *fast* feature. Finally, we multiply *p* by 0.2 and end up with 0.0324 -- the probability that the beaver is cute.

You will be given a decision tree and a list of animals with their features. For each item, you need to return the probability that the animal is cute.

Input

The first line of input contains a single integer, **N**, the number of test cases. **N** test cases follow.

Each test case description will start with a line that contains an integer **L** -- the number of lines that describe a decision tree. The next **L** lines will contain a decision tree in the format described above. The line after that will contain **A** -- the number of animals. The next **A** lines will each contain the description of one animal in the following format.

$animal\ \mathbf{n}\ feature_1\ feature_2\ \ldots\ feature_n$

Output

For each test case, output one line containing "Case #**x**:" followed by exactly **A** lines, one per animal, in the same order as they appear in the input. Each line should contain the probability that the animal is cute. Answers that are precise to within an absolute or relative error of 10^{-6} will be considered correct.

Limits

1 ≤ **N** ≤ 100
All weights will be between 0 and 1, inclusive.
All weights will consist of only digits with at most one decimal point.
The weights will not start or end with a decimal point.
The weights will not have more than one 0 before a decimal point.
All animals and features will consist of between 1 and 10 lower case English letters.
All animal names within a test case will be distinct.
All feature names for a single animal will be distinct.
Each of the **L** lines in a decision tree definition will have at most 80 characters, not including the endlines.

Small dataset

1 ≤ **L** ≤ 10
1 ≤ **A** ≤ 10
0 ≤ **n** ≤ 5

Large dataset

1 ≤ **L** ≤ 100
1 ≤ **A** ≤ 100
0 ≤ **n** ≤ 100

Sample

Input

```
2
3
(0.5 cool
  ( 1.000)
  (0.5 ))
2
anteater 1 cool
cockroach 0
13
(0.2 furry
  (0.81 fast
    (0.3)
    (0.2)
  )
  (0.1 fishy
    (0.3 freshwater
      (0.01)
      (0.01)
    )
    (0.1)
  )
)
3
beaver 2 furry freshwater
trout 4 fast freshwater fishy rainbowy
dodo 1 extinct
```

Output

```
Case #1:
0.5000000
0.2500000
Case #2:
0.0324000
0.0000600
0.0020000
```

Submissions

Decision Tree

10pt Not attempted **1512/1752 users** correct (86%)

11pt Not attempted **1266/1544 users** correct (82%)

The Next Number

9pt Not attempted **2559/3329 users** correct (77%)

26pt Not attempted **1890/2557 users** correct (74%)

Square Math

12pt Not attempted **157/422 users** correct (37%)

32pt Not attempted **69/168 users** correct (41%)

Top Scores

ACRush	100
ftc	100
bmerry	100
andrewzta	100
ipknHama	100
halyavin	100
mystic	100
Yarin	100
Khuc.Anh.Tuan	100
dgozman	100

Problem B. The Next Number

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 26 points — Solve B-large

Problem

You are writing out a list of numbers. Your list contains all numbers with exactly $\mathbf{D_i}$ digits in its decimal representation which are equal to **i**, for each **i** between 1 and 9, inclusive. You are writing them out in ascending order.

For example, you might be writing every number with two '1's and one '5'. Your list would begin 115, 151, 511, 1015, 1051.

Given **N**, the last number you wrote, compute what the next number in the list will be.

Input

The first line of input contains an integer **T**, the number of test cases in the input. **T** lines follow, one for each test case, each containing a single integer **N**.

Output

For each test case, output

```
Case #X: K
```

where **X** is the test case number, starting from 1, and **K** is the next integer in the list.

Limits

Small dataset

$1 \le T \le 50$
$1 \le N \le 10^6$

Large dataset

$1 \le T \le 500$
$1 \le N \le 10^{20}$

Sample

Input	Output
3	
115	Case #1: 151
1051	Case #2: 1105
6233	Case #3: 6323

Problem C. Square Math

Problem

Say we have a square that has **W** cells on each side and, therefore, $\mathbf{W}^2$ cells total. Let's go further and fill each cell with one of the following:

- A digit from 0 to 9;
- The addition sign (+);
- The subtraction sign (-).

If, finally, we add a constraint that no 2 digits are horizontally or vertically adjacent and no 2 operators (+ or -) are horizontally or vertically adjacent, then our square can be called an "arithmetic square".

Square Math is the name of a puzzle where, given an arithmetic square, we start from any numeric cell and move either horizontally or vertically a cell at a time, finally ending in a numerical cell. The mathematical expression we get from the traversal is evaluated to get a single value. For example:

```
2+3
+4-
1+0
```

The above is a valid arithmetic square of size **W** = 3. If we start from "2", move horizontally right, then vertically down, we'll get "2+4", which gives a value of "6". If we further move horizontally right, then vertically up, we'll get "2+4-3", which is equal to "3".

In Square Math, there is no limit to how many times you can use a particular cell. It is perfectly legal to move from a cell to its neighbor, then back to the original cell. Given an arithmetic square and a list of queries, your task is to find a Square Math expression which evaluates to each query.

Input

The first line of input contains a single integer, **T**. **T** test cases follow. The first line of each test case contains 2 integers, **W** and **Q**. **W** lines follow, each containing **W** characters, representing the arithmetic square. Don't worry, all arithmetic squares in the input are well-formed. The following line contains a space separated list of **Q** integers, representing the values which need to be computed by using Square Math (the queries). You can assume that all given values will have at least one possible Square Math solution.

Output

For each test case, begin output with "Case #**X**:" on a line by itself, where **X** is the test case number, starting from 1. Then, for each query within the test case, print the Square Math expression which evaluates to the query on a line by itself.

In the case where there are multiple possible Square Math expressions, print the one that is shortest. If there is still a tie, print the lexicographically smallest expression. Remember that '+' is lexicographically smaller than '-'.

Limits

$1 \le \mathbf{T} \le 60$

Small dataset

$2 \le \mathbf{W} \le 10$
$1 \le \mathbf{Q} \le 20$
$1 \le$ each query ≤ 50

Large dataset

$2 \le \mathbf{W} \le 20$
$1 \le \mathbf{Q} \le 50$
$1 \le$ each query ≤ 250

Sample

Input	Output

<pre>2
5 3
2+1-2
+3-4+
5+2+1
-4-0-
9+5+1
20 30 40
3 2
2+1
+4+
5+1
2 20</pre>
	<pre>Case #1:
	1+5+5+9
	3+4+5+9+9
	4+9+9+9+9
	Case #2:
	2
	5+5+5+5</pre>

Problem A. All Your Base

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

In A.D. 2100, aliens came to Earth. They wrote a message in a cryptic language, and next to it they wrote a series of symbols. We've come to the conclusion that the symbols indicate a number: the number of seconds before war begins!

Unfortunately we have no idea what each symbol means. We've decided that each symbol indicates one digit, but we aren't sure what each digit means or what base the aliens are using. For example, if they wrote "ab2ac999", they could have meant "31536000" in base 10 -- exactly one year -- or they could have meant "12314555" in base 6 -- 398951 seconds, or about four and a half days. We are sure of three things: the number is positive; like us, the aliens will never start a number with a zero; and they aren't using unary (base 1).

Your job is to determine the minimum possible number of seconds before war begins.

Input

The first line of input contains a single integer, **T**. **T** test cases follow. Each test case is a string on a line by itself. The line will contain only characters in the 'a' to 'z' and '0' to '9' ranges (with no spaces and no punctuation), representing the message the aliens left us. The test cases are independent, and can be in different bases with the symbols meaning different things.

Output

For each test case, output a line in the following format:

```
Case #X: V
```

Where **X** is the case number (starting from 1) and **V** is the minimum number of seconds before war begins.

Limits

$1 \leq \mathbf{T} \leq 100$
The answer will never exceed 10^{18}

Small dataset

$1 \leq$ the length of each line < 10

Large dataset

$1 \leq$ the length of each line < 61

Sample

```
Input        Output

3            Case #1: 201
11001001     Case #2: 75
cats         Case #3: 11
zig
```

Submissions

All Your Base

8pt	Not attempted	**2176/2473 users** correct (88%)
15pt	Not attempted	**1441/2203 users** correct (65%)

Center of Mass

10pt	Not attempted	**823/1428 users** correct (58%)
17pt	Not attempted	**737/913 users** correct (81%)

Bribe the Prisoners

15pt	Not attempted	**1061/1579 users** correct (67%)
35pt	Not attempted	**302/735 users** correct (41%)

Top Scores

tikitikirevenge	100
Progbeat	100
Zeroline	100
maojm	100
WSX	100
Onufry	100
Imba	100
ZhukovDmitry	100
Al.Cash	100
Ostap	100

Problem B. Center of Mass

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 B-small
Large input 17 points	Solve B-large

Problem

You are studying a swarm of **N** fireflies. Each firefly is moving in a straight line at a constant speed. You are standing at the center of the universe, at position (0, 0, 0). Each firefly has the same mass, and you want to know how close the center of the swarm will get to your location (the origin).

You know the position and velocity of each firefly at t = 0, and are only interested in t ≥ 0. The fireflies have constant velocity, and may pass freely through all of space, including each other and you. Let M(t) be the location of the center of mass of the **N** fireflies at time t. Let d(t) be the distance between your position and M(t) at time t. Find the minimum value of d(t), d_{min}, and the earliest time when d(t) = d_{min}, t_{min}.

Input

The first line of input contains a single integer **T**, the number of test cases. Each test case starts with a line that contains an integer **N**, the number of fireflies, followed by **N** lines of the form

```
x y z vx vy vz
```

Each of these lines describes one firefly: (x, y, z) is its initial position at time t = 0, and (vx, vy, vz) is its velocity.

Output

For each test case, output

```
Case #X: dmin tmin
```

where **X** is the test case number, starting from 1. Any answer with absolute or relative error of at most 10^{-5} will be accepted.

Limits

All the numbers in the input will be integers.
1 ≤ **T** ≤ 100
The values of x, y, z, vx, vy and vz will be between -5000 and 5000, inclusive.

Small dataset

3 ≤ **N** ≤ 10

Large dataset

3 ≤ **N** ≤ 500

Sample

Input	Output

<pre>3
3
3 0 -4 0 0 3
-3 -2 -1 3 0 0
-3 -1 2 0 3 0
3
-5 0 0 1 0 0
-7 0 0 1 0 0
-6 3 0 1 0 0
4
1 2 3 1 2 3
3 2 1 3 2 1
1 0 0 0 0 -1
0 10 0 0 -10 -1</pre>
	<pre>Case #1: 0.00000000 1.00000000
	Case #2: 1.00000000 6.00000000
	Case #3: 3.36340601 1.00000000</pre>

Notes

Given **N** points (x_i, y_i, z_i), their center of the mass is the point (x_c, y_c, z_c), where:

$$x_c = (x_1 + x_2 + ... + x_N) / N$$
$$y_c = (y_1 + y_2 + ... + y_N) / N$$
$$z_c = (z_1 + z_2 + ... + z_N) / N$$

Problem C. **Bribe the Prisoners**

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 15 points	Solve C-small
Large input 35 points	Solve C-large

Problem

In a kingdom there are prison cells (numbered 1 to **P**) built to form a straight line segment. Cells number **i** and **i+1** are adjacent, and prisoners in adjacent cells are called "neighbours." A wall with a window separates adjacent cells, and neighbours can communicate through that window.

All prisoners live in peace until a prisoner is released. When that happens, the released prisoner's neighbours find out, and each communicates this to his other neighbour. That prisoner passes it on to *his* other neighbour, and so on until they reach a prisoner with no other neighbour (because he is in cell 1, or in cell **P**, or the other adjacent cell is empty). A prisoner who discovers that another prisoner has been released will angrily break everything in his cell, unless he is bribed with a gold coin. So, after releasing a prisoner in cell **A**, all prisoners housed on either side of cell **A** - until cell 1, cell **P** or an empty cell - need to be bribed.

Assume that each prison cell is initially occupied by exactly one prisoner, and that only one prisoner can be released per day. Given the list of **Q** prisoners to be released in **Q** days, find the minimum total number of gold coins needed as bribes if the prisoners may be released in any order.

Note that each bribe only has an effect for one day. If a prisoner who was bribed yesterday hears about another released prisoner today, then he needs to be bribed again.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each case consists of 2 lines. The first line is formatted as

```
P Q
```

where **P** is the number of prison cells and **Q** is the number of prisoners to be released.
This will be followed by a line with **Q** distinct cell numbers (of the prisoners to be released), space separated, sorted in ascending order.

Output

For each test case, output one line in the format

```
Case #X: C
```

where **X** is the case number, starting from 1, and **C** is the minimum number of gold coins needed as bribes.

Limits

$1 \le \mathbf{N} \le 100$
$\mathbf{Q} \le \mathbf{P}$
Each cell number is between 1 and **P**, inclusive.

Small dataset

$1 \le \mathbf{P} \le 100$
$1 \le \mathbf{Q} \le 5$

Large dataset

$1 \le \mathbf{P} \le 10000$
$1 \le \mathbf{Q} \le 100$

Sample

Input	Output
2 8 1	Case #1: 7 Case #2: 35

```
3
20 3
3 6 14
```

Note

In the second sample case, you first release the person in cell 14, then cell 6, then cell 3. The number of gold coins needed is 19 + 12 + 4 = 35. If you instead release the person in cell 6 first, the cost will be 19 + 4 + 13 = 36.

Problem A. Crazy Rows

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 A-small
Large input 10 points	Solve A-large

Problem

You are given an **N** x **N** matrix with 0 and 1 values. You can swap any two *adjacent* rows of the matrix.

Your goal is to have all the 1 values in the matrix below or on the main diagonal. That is, for each X where 1 ≤ X ≤ N, there must be no 1 values in row X that are to the right of column X.

Return the minimum number of row swaps you need to achieve the goal.

Input

The first line of input gives the number of cases, **T**. **T** test cases follow.
The first line of each test case has one integer, **N**. Each of the next **N** lines contains **N** characters. Each character is either 0 or 1.

Output

For each test case, output

```
Case #X: K
```

where **X** is the test case number, starting from 1, and **K** is the minimum number of row swaps needed to have all the 1 values in the matrix below or on the main diagonal.

You are guaranteed that there is a solution for each test case.

Limits

1 ≤ **T** ≤ 60

Small dataset

1 ≤ **N** ≤ 8

Large dataset

1 ≤ **N** ≤ 40

Sample

```
Input     Output

3         Case #1: 0
2         Case #2: 2
10        Case #3: 4
11
3
001
100
010
4
1110
1100
1100
1000
```


Top Scores

ACRush	100
winger	100
iwi	100
wata	100
bwps	100
natalia	100
Burunduk1	100
AS1	100
Khuc.Anh.Tuan	100
Nerevar	100

Submissions

Crazy Rows
6pt Not attempted **1837/2092 users** correct (88%)
10pt Not attempted **1605/1744 users** correct (92%)

A Digging Problem
9pt Not attempted **193/388 users** correct (50%)
17pt Not attempted **70/152 users** correct (46%)

Stock Charts
7pt Not attempted **741/1384 users** correct (54%)
21pt Not attempted **355/537 users** correct (66%)

Watering Plants
5pt Not attempted **1251/1420 users** correct (88%)
25pt Not attempted **64/226 users** correct (28%)

Top Scores

ACRush	100
winger	100
iwi	100
wata	100
bwps	100
natalia	100
Burunduk1	100
AS1	100
Khuc.Anh.Tuan	100
Nerevar	100

Problem B. **A Digging Problem**

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 17 points	Solve B-large

Problem

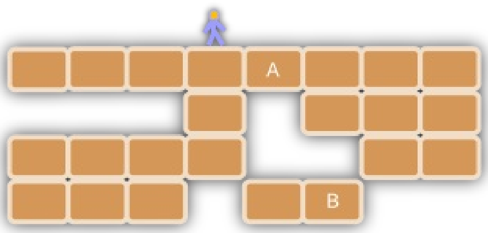
The cave is on fire and there is smoke everywhere! You are trying to dig your way to the bottom of the cave where you can breathe. The problem is that there are some air holes inside the cave, and you don't want to fall too much or you will get hurt.

The cave is represented as an **R** x **C** matrix with air holes and solid rock cells. You start at position (1, 1), which is in the top-left corner. You can move one cell at a time, left or right, if that cell is empty (an air hole). After moving, if the cell below is empty, you fall down until you hit solid rock or the bottom of the cave. The falling distance must be at most **F**,or you will get hurt. You must reach the bottom of the cave without getting hurt. While falling you cannot move left or right.

You can also "dig", turning a cell that contains solid rock into an air hole. The cell that you dig can be one of two cells: the one to your right and below, or the one to your left and below. The cell above the one you are digging has to be empty. While falling you cannot dig.

Your goal is not only to get to the bottom of cave, but also to "dig" as few cells as possible.

Let's describe the operations with a concrete example:



You start at (1, 1) and move right 3 times to position (1, 4), just like the picture.
You dig the rock at position (2, 5). Cell "A" becomes empty.
You move right one position and since there is no cell below you fall 3 cells to position (4, 5). You dig the rock at position (5, 6). Cell "B" becomes empty.
You move right one position and since there is no cell below you fall 1 cell to position (5, 6).
You have reached the bottom of the cave by digging 2 cells.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. The first line of each case is formatted as

```
R C F
```

where **R** is the number of rows in the cave, **C** is the number of columns in the cave, and **F** is the maximum distance you can fall without getting hurt.
This is followed by **R**, rows each of which contains **C** characters. Each character can be one of two things:

- # for a solid rock
- . for an air hole

The top-left cell will always be empty, and the cell below it will be a solid rock.

Output

For each test case, output one line in the format

```
Case #X: No/Yes [D]
```

where **X** is the case number, starting from 1. Output "No" if you cannot reach the bottom of the cave. Output "Yes **D**" if the bottom of the cave can be reached and the minimum number of cells that need digging is **D**.

Limits

1 ≤ **N** ≤ 50
1 ≤ **F** < **R**

Small dataset

2 ≤ **R** ≤ 10
2 ≤ **C** ≤ 6

Large dataset

2 ≤ **R** ≤ 50
2 ≤ **C** ≤ 50

Sample

Input	Output

<pre>4
5 8 3
....
########
...#.###
####..##
###.##..
2 2 1
.#
##
3 3 1
...
###
###
3 2 1
..
#.
..</pre>
	<pre>Case #1: Yes 2
	Case #2: No
	Case #3: Yes 3
	Case #4: No</pre>

Submissions

Crazy Rows
6pt Not attempted **1837/2092 users** correct (88%)
10pt Not attempted **1605/1744 users** correct (92%)

A Digging Problem
9pt Not attempted **193/388 users** correct (50%)
17pt Not attempted **70/152 users** correct (46%)

Stock Charts
7pt Not attempted **741/1384 users** correct (54%)
21pt Not attempted **355/537 users** correct (66%)

Watering Plants
5pt Not attempted **1251/1420 users** correct (88%)
25pt Not attempted **64/226 users** correct (28%)

Top Scores

ACRush	100
winger	100
iwi	100
wata	100
bwps	100
natalia	100
Burunduk1	100
AS1	100
Khuc.Anh.Tuan	100
Nerevar	100

Problem C. Stock Charts

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
7 points
Solve C-small

Large input
21 points
Solve C-large

Problem

You're in the middle of writing your newspaper's end-of-year economics summary, and you've decided that you want to show a number of charts to demonstrate how different stocks have performed over the course of the last year. You've already decided that you want to show the price of **n** different stocks, all at the same **k** points of the year.

A *simple chart* of one stock's price would draw lines between the points (0, $price_0$), (1, $price_1$), ... , (k-1, $price_{k-1}$), where $price_i$ is the price of the stock at the *i*th point in time.

In order to save space, you have invented the concept of an *overlaid chart*. An overlaid chart is the combination of one or more simple charts, and shows the prices of multiple stocks (simply drawing a line for each one). In order to avoid confusion between the stocks shown in a chart, the lines in an overlaid chart may not cross or touch.

Given a list of *n* stocks' prices at each of *k* time points, determine the minimum number of overlaid charts you need to show all of the stocks' prices.

Input

The first line of input will contain a single integer **T**, the number of test cases. After this will follow **T** test cases on different lines, each of the form:

```
n k
price0,0 price0,1 ... price0,k-1
price1,0 price1,1 ... price1,k-1
...
pricen-1,0 pricen-1,1 ... pricen-1,k-1
```

Where $price_{i,j}$ is an integer, the price of the *i*th stock at time *j*.

Output

For each test case, a single line containing "Case #X: Y", where *X* is the number of the test-case (1-indexed) and *Y* is the minimum number of overlaid charts needed to show the prices of all of the stocks.

Limits

1 ≤ **T** ≤ 100
2 ≤ **k** ≤ 25
0 ≤ $price_{i,j}$ ≤ 1000000

Small Input

1 ≤ **n** ≤ 16

Large Input

1 ≤ **n** ≤ 100

Sample

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

```
5 4
4 4
4 1
```

Submissions

Crazy Rows

6pt Not attempted **1837/2092 users** correct (88%)
10pt Not attempted **1605/1744 users** correct (92%)

A Digging Problem

9pt Not attempted **193/388 users** correct (50%)
17pt Not attempted **70/152 users** correct (46%)

Stock Charts

7pt Not attempted **741/1384 users** correct (54%)
21pt Not attempted **355/537 users** correct (66%)

Watering Plants

5pt Not attempted **1251/1420 users** correct (88%)
25pt Not attempted **64/226 users** correct (28%)

Top Scores

ACRush	100
winger	100
iwi	100
wata	100
bwps	100
natalia	100
Burunduk1	100
AS1	100
Khuc.Anh.Tuan	100
Nerevar	100

Problem D. Watering Plants

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 5 points — Solve D-small

Large input 25 points — Solve D-large

Problem

In your greenhouse you have a number of plants which you need to water.

Each of the plants takes up an area which is a circle. No two of the plants overlap or touch each other.

You are going to buy two sprinklers. Each of the sprinklers will spray everything within a circle of radius **R** with water.

One of the sprinklers will run in the morning, and one will run at night. For you to be satisfied that a plant will get enough water, either the whole area of the plant must be watered in the morning, or the the whole area of the plant must be watered at night. So each of the circles representing a plant must be completely in one or both of the two circles representing the area the sprinklers can water.

Given the location and radius of each of the plants, find the minimum radius **R** so that it is possible to place the two sprinklers to water all the plants. The sprinklers will be installed on the ceiling, so a sprinkler's position can be inside the area of a plant.

Input

- One line containing an integer **C**, the number of test cases in the input file.

For each test case, there will be:

- One line containing **N**, where **N** is the number of plants you have.
- **N** lines, one for each plant, containing three integers "**X Y R**", where (**X**, **Y**) are the coordinates of the center of the plant, and **R** is the radius of the plant.

Output

For each test case:

- One line containing the string "Case #x: R" where *x* is the number of the test case, starting from 1, and **R** is the minimum radius of the sprinklers.

Any answer with absolute or relative error of at most 10^{-5} will be accepted.

Limits

All numbers in the input file are integers.
1 ≤ **X** ≤ 1000
1 ≤ **Y** ≤ 1000
1 ≤ **R** ≤ 100

Small Input

1 ≤ **C** ≤ 10
1 ≤ **N** ≤ 3

Large Input

1 ≤ **C** ≤ 30
1 ≤ **N** ≤ 40

Sample

Input	Output

`5`
`3`
`20 10 2`
`20 20 2`
`40 10 3`
`3`
`20 10 3`
 	
`30 10 3`
`40 10 3`
	`Case #1: 7.000000`
	`Case #2: 8.000000`
	`Case #3: 26.000000`
	`Case #4: 8.071067`
	`Case #5: 51`

```
5
100 100 1
140 100 1
100 130 1
100 500 1
150 500 1
8
100 100 1
110 100 1
100 110 1
110 110 1
200 200 1
210 200 1
200 210 1
210 210 1
4
100 100 1
200 100 1
200 103 1
300 103 1
```

In the first case, a sprinkler of radius at least 7 centered at (20,15) will water the first two plants. A sprinkler of radius at least 3 will water the plant at (40,10).

In the second case, one of the two sprinklers will need a radius of at least 8. Note that the plant at (30,10) must be covered entirely by one of the two sprinklers.

Round 3 2009

A. EZ-Sokoban

B. Alphabetomials

C. Football Team

D. Interesting Ranges

Contest Analysis

Questions asked

Submissions

EZ-Sokoban

7pt Not attempted **231/262 users** correct (88%)

10pt Not attempted **158/219 users** correct (72%)

Alphabetomials

4pt Not attempted **186/225 users** correct (83%)

20pt Not attempted **37/71 users** correct (52%)

Football Team

8pt Not attempted **36/138 users** correct (26%)

19pt Not attempted **16/36 users** correct (44%)

Interesting Ranges

9pt Not attempted **24/41 users** correct (59%)

23pt Not attempted **1/3 users** correct (33%)

Top Scores

bmerry	77
qizichao	77
winger	68
Ahyangyi	68
misof	50
rem	50
kia	50
mystic	50
marek.cygan	50
dzhulgakov	50

Problem A. **EZ-Sokoban**

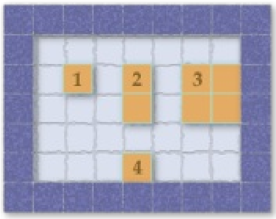
This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 7 points	Solve A-small
Large input 10 points	Solve A-large

Problem

Sokoban is a famous Japanese puzzle game. Sokoban is Japanese for "warehouse keeper". In this game, your goal is to push boxes to their designated locations in the warehouse. To push a box, the area behind the box and in front of the box must be empty. This is because you stand behind the box when pushing and you can push only one box at a time. You cannot push a box out of the board and you cannot stand outside the board when pushing a box.

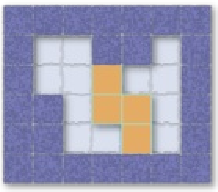
For example, in this picture:



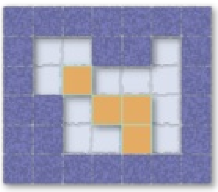
Box 1 can be pushed in any of the four directions because the four spaces adjacent to it are empty. Box 2 can only be pushed east or west; it cannot be pushed north or south because the space to its south is not empty. Box 3 cannot be pushed in any direction. Box 4 can only be pushed east or west because there is a wall south of it.

Sokoban was proved to be a P-Space complete problem, but we deal with an easier variation here. In our variation of Sokoban, boxes have strong magnets inside and they have to stick together *almost* all the time. Under "stable" conditions, all boxes should be connected, edge to edge. This means that from any box we can get to any other box by going through boxes that share an edge. If you push a box and boxes are no longer connected, you are in "dangerous mode". In dangerous mode, the next push must make the boxes connected again.

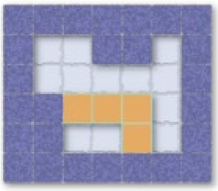
For example, in this picture:



The situation is stable, since all 4 boxes are connected, edge to edge. Let's assume that you decided to push the northmost box west:



Now, we are in dangerous mode since the northmost box is not connected to any other boxes. The next push must return us to a stable position. For example, we can push that northmost box south:



Making the boxes stable again.

A Sokoban puzzle consists of a board, initial configuration of the boxes and the final configuration (where we want the boxes to be at the end). Given an EZ-Sokoban puzzle, find a solution that makes the minimum number of box moves, or decide that it can't be solved. The final and initial configurations will not be in "dangerous" mode.

To simplify things, we will assume that you, the warehouse keeper, can jump at any time to any empty spot on the board.

Input

The first line in the input file contains the number of cases, **T**.

Each case consists of several lines. The first line contains **R** and **C**, the number of rows and columns of the board, separated by one space. This is followed by **R** lines. Each line contains **C** characters describing the board:

- '.' is an empty spot
- '#' is a wall
- 'x' is a goal (where a box should be at the end)
- 'o' is a box
- 'w' is a both a box and a goal

The number of boxes will be equal to the number of goals.

Output

For each test case, output

```
Case #X: K
```

where **X** is the test case number, starting from 1, and **K** is the minimum number of box moves that are needed to solve the puzzle or **-1** if it cannot be solved.

Limits

1 ≤ **T** ≤ 50
1 ≤ **R**,**C** ≤ 12

Small dataset

1 ≤ the number of boxes ≤ 2

Large dataset

1 ≤ the number of boxes ≤ 5

Sample

Input	Output

<pre>6
5 4
....
#..#
#xx#
#oo#
#..#
7 7
.######
.x....#
.x....#
..#oo.#
..#...#
.######
.######
7 7
#######
#x....#
#xx.o.#
#..oo.#
#.#...#
#######
#######
4 10</pre>
	<pre>Case #1: 2
	Case #2: 8
	Case #3: 12
	Case #4: 8
	Case #5: -1
	Case #6: 2</pre>

```
##########
#.x...o..#
#.x...o..#
##########
7 7
#######
#x....#
#x..o.#
#x#oo.#
#.#...#
#######
#######
3 4
.#x.
.oww
....
```

Problem B. Alphabetomials

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 4 points	Solve B-small
Large input 20 points	Solve B-large

Problem

As we all know, there is a big difference between polynomials of degree 4 and those of degree 5. The question of the non-existence of a closed formula for the roots of general degree 5 polynomials produced the famous Galois theory, which, as far as the author sees, bears no relation to our problem here.

We consider only the multi-variable polynomials of degree up to 4, over 26 variables, represented by the set of 26 lowercase English letters. Here is one such polynomial:

```
aber+aab+c
```

Given a string *s*, we evaluate the polynomial on it. The evaluation gives *p*(*S*) as follows: Each variable is substituted with the number of appearances of that letter in *S*.
For example, take the polynomial above, and let *S* = "abracadabra edgar". There are six a's, two b's, one c, one e, and three r's. So

```
p(S) = 6 * 2 * 1 * 3 + 6 * 6 * 2 + 1 = 109.
```

Given a dictionary of distinct words that consist of only lower case letters, we call a string *S* a *d-phrase* if

$$S = "S_1\ S_2\ S_3\ \ldots\ S_d",$$

where S_i is any word in the dictionary, for 1 ≤ i ≤ d. i.e., *S* is in the form of *d* dictionary words separated with spaces. Given a number **K** ≤ 10, your task is, for each 1≤ *d* ≤ **K**, to compute the sum of *p*(*S*) over all the *d*-phrases. Since the answers might be big, you are asked to compute the remainder when the answer is divided by 10009.

Input

The first line contains the number of cases **T**. **T** test cases follow. The format of each test case is:
A line containing an expression *p* for the multi-variable polynomial, as described below in this section, then a space, then follows an integer **K**.
A line with an integer **n**, the number of words in the dictionary.
Then **n** lines, each with a word, consists of only lower case letters. No word will be repeated in the same test case.

We always write a polynomial in the form of a sum of terms; each term is a product of variables. We write a^t simply as *t* *a*'s concatenated together. For example, a^2b is written as *aab*. Variables in each term are always lexicographically non-decreasing.

Output

For each test case, output a single line in the form

$$\text{Case \#X: } sum_1\ sum_2\ \ldots\ sum_K$$

where *X* is the case number starting from 1, and sum_i is the sum of *p*(*S*), where *S* ranges over all i-phrases, modulo 10009.

Limits

1 ≤ **T** ≤ 100.
The string *p* consists of one or more terms joined by '+'. It will not start nor end with a '+'. There will be at most 5 terms for each *p*. Each term consists at least 1 and at most 4 lower case letters, sorted in non-decreasing order. No two terms in the same polynomial will be the same.
Each word is non-empty, consists only of lower case English letters, and will not be longer than 50 characters. No word will be repeated in the same

dictionary.

Small dataset

1 ≤ **n** ≤ 20
1 ≤ **K** ≤ 5

Large dataset

1 ≤ **n** ≤ 100
1 ≤ **K** ≤ 10

Sample

Input	Output

<pre>2
ehw+hwww 5
6
where
when
what
whether
who
whose
a+e+i+o+u 3
4
apple
orange
watermelon
banana</pre>
	<pre>Case #1: 15 1032 7522 6864 253
	Case #2: 12 96 576</pre>

Problem C. **Football Team**

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 C-small
Large input 19 points	Solve C-large

Problem

A football team will be standing in rows to have a photograph taken. The location of each player will be given by two integers x and y, where y gives the number of the row, and x gives the distance of the player from the left edge of the row. The x values will be all different.

In order to make the photo more interesting, you're going to make sure players who are near each other have shirts of different colors. To do this, you set the following rule:
For each player P:

- The closest player to the right of P in the same row, if there is such a player, must have a different shirt color.
- The closest player to the right of P in the previous row, if there is such a player, must have a different shirt color.
- The closest player to the right of P in the next row, if there is such a player, must have a different shirt color.

More formally, if there is a player at (x1,y1) and (x2,y2), where x1<x2, then those two players must have different shirt colors if:

- y1 - 1 ≤ y2 ≤ y1 + 1, and
- there is no x3 such that there is a player at (x3, y2) and x1 < x3 < x2.

Find the minimum number of distinct shirt colors required so that this is possible.

Input

The first line of input contains a single integer **T**, the number of test cases. Each test case starts with a line that contains an integer **N**, the number of players, followed by **N** lines of the form

```
x y
```

each specifying the position of one player.

Output

For each test case, output

```
Case #X: c
```

where **X** is the test case number, starting from 1, and **c** is the minimum number of colors required.

Limits

1 ≤ **T** ≤ 100
1 ≤ **x** ≤ 1000
The values of x will all be different.

Small dataset

1 ≤ **y** ≤ 15
1 ≤ **N** ≤ 100

Large dataset

1 ≤ **y** ≤ 30
1 ≤ **N** ≤ 1000

Sample

Input	Output
3 3	Case #1: 1 Case #2: 2

```
10 10    Case #3: 3
8 15
12 7
5
1 1
2 1
3 1
4 1
5 1
3
1 1
2 2
3 1
```

Round 3 2009

A. EZ-Sokoban
B. Alphabetomials
C. Football Team
D. Interesting Ranges

Contest Analysis
Questions asked

Submissions

EZ-Sokoban

7pt	Not attempted **231/262 users** correct (88%)
10pt	Not attempted **158/219 users** correct (72%)

Alphabetomials

4pt	Not attempted **186/225 users** correct (83%)
20pt	Not attempted **37/71 users** correct (52%)

Football Team

8pt	Not attempted **36/138 users** correct (26%)
19pt	Not attempted **16/36 users** correct (44%)

Interesting Ranges

9pt	Not attempted **24/41 users** correct (59%)
23pt	Not attempted **1/3 users** correct (33%)

Top Scores

bmerry	77
qizichao	77
winger	68
Ahyangyi	68
misof	50
rem	50
kia	50
mystic	50
marek.cygan	50
dzhulgakov	50

Problem D. **Interesting Ranges**

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 D-small
Large input 23 points	Solve D-large

Problem

A positive integer is a *palindrome* if its decimal representation (without leading zeros) is a palindromic string (a string that reads the same forwards and backwards). For example, the numbers 5, 77, 363, 4884, 11111, 12121 and 349943 are palindromes.

A range of integers is *interesting* if it contains an even number of palindromes. The range [L, R], with L ≤ R, is defined as the sequence of integers from L to R (inclusive): (L, L+1, L+2, ..., R-1, R). L and R are the range's first and last numbers.

The range $[L_1,R_1]$ is a *subrange* of [L,R] if $L \le L_1 \le R_1 \le R$. Your job is to determine how many interesting subranges of [L,R] there are.

Input

The first line of input gives the number of test cases, **T**. **T** test cases follow. Each test case is a single line containing two positive integers, **L** and **R** (in that order), separated by a space.

Output

For each test case, output one line. That line should contain "Case #x: y", where x is the case number starting with 1, and y is the number of interesting subranges of [L,R], modulo 1000000007.

Limits

$1 \le \mathbf{T} \le 120$

Small dataset

$1 \le \mathbf{L} \le \mathbf{R} \le 10^{13}$

Large dataset

$1 \le \mathbf{L} \le \mathbf{R} \le 10^{100}$

Sample

Input	Output
3 1 2 1 7 12 110	Case #1: 1 Case #2: 12 Case #3: 2466

World Finals 2009

A. Year of More Code Jam
B. Min Perimeter
C. Doubly-sorted Grid
D. Wi-fi Towers
E. Marbles
F. Lights

Contest Analysis

Questions asked 1

Submissions

Year of More Code Jam

5pt	Not attempted **16/17 users** correct (94%)
12pt	Not attempted **9/15 users** correct (60%)

Min Perimeter

5pt	Not attempted **17/19 users** correct (89%)
15pt	Not attempted **4/13 users** correct (31%)

Doubly-sorted Grid

10pt	Not attempted **16/16 users** correct (100%)
20pt	Not attempted **4/5 users** correct (80%)

Wi-fi Towers

3pt	Not attempted **22/22 users** correct (100%)
25pt	Not attempted **9/12 users** correct (75%)

Marbles

7pt	Not attempted **16/19 users** correct (84%)
32pt	Not attempted **2/8 users** correct (25%)

Lights

21pt	Not attempted **2/4 users** correct (50%)
45pt	Not attempted **1/2 users** correct (50%)

Top Scores

ACRush	168
qizichao	87
wata	81
ZhukovDmitry	70
dzhulgakov	69
nika	62
Vitaliy	62
kalinov	55
halyavin	54
bmerry	50

Problem A. **Year of More Code Jam**

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 5 points	Solve A-small
Large input 12 points	Solve A-large

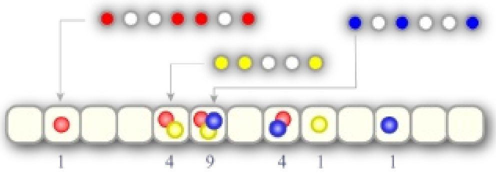
Problem

A new year brings a new calendar, new challenges, and a lot of new fun in life. Some things, however, never change. There are still many great programming contests to be held, and our heroine Sphinny's passion for them remains unchanged.

There are several tournaments Sphinny is interested in. Each tournament will consist of a number of rounds. The organizer of each tournament has not decided on what date the tournament will start, but has decided how many rounds there will be in the tournament and how many days after the start date each round will be.

In some situations, two or more rounds (from different tournaments) can be scheduled on the same day. As Sphinny is so keen on problem solving, she will be happier if more rounds are scheduled on the same day. Her happiness value is computed as follows: for each day on which there are *S* rounds, her happiness will be increased by S^2. Her happiness starts at 0 (don't worry — 0 is a happy place to start).

In the picture below there are three tournaments, each represented by a different color, and Sphinny's total happiness is 20. One tournament starts on the second day of the year, one starts on the fifth day of the year, and one starts on the sixth day of the year.



There are **N** days in the year. Each tournament will begin on any of the **N** days with equal probability. The big question for this year is what the expected value of Sphinny's happiness is.

As a perfectionist, she is not going to solve the problem approximately. Instead, she wants to know the result exactly. The number of tournaments is **T**, and there are $\mathbf{N}^\mathbf{T}$ equally likely ways to select the start dates of the tournaments. She is going to express her expected happiness as *K*+*A*/*B*, where *K* and *B* are positive integers and *A* is a non-negative integer less than *B*. If *A* is zero then *B* must be one, otherwise *A* and *B* must not have a common factor greater than one.

If a tournament starts late enough in the year, some of its rounds might be scheduled during the next year. Those rounds do not contribute to Sphinny's happiness this year.

Input

The first line of the input is a single integer **C**, the number of test cases. **C** tests follow. The first line of each test case is in the form

```
N T
```

where **N** is the number of days in the year, and **T** is the number of tournaments. **T** lines then follow, one for each tournament, in the format

```
m d2 d3 ... dm
```

indicating that there are **m** rounds, and the i-th round will be held on day $\mathbf{d}_i$ of the tournament. The first round of a tournament is held on day 1 ($\mathbf{d}_1$ = 1).

Output

For each test, output one line of the form

```
Case #X: K+A/B
```

where X is the case number, starting from 1, and K, A and B are as described above.

Limits

$1 \le \mathbf{C} \le 50$
$1 \le \mathbf{N} \le 10^9$
$2 \le \mathbf{m} \le 50$
$1 < \mathbf{d}_2 < \mathbf{d}_3 < \dots < \mathbf{d}_m \le 10000$

Small dataset

$1 \le \mathbf{T} \le 2$

Large dataset

$1 \le \mathbf{T} \le 50$

Sample

Input	Output
3 1 1 2 2 4 2 3 2 4 2 3 10 3 4 2 5 9 3 4 8 5 2 3 9 12	Case #1: 1+0/1 Case #2: 5+1/8 Case #3: 11+19/25

World Finals 2009

Submissions

Year of More Code Jam

5pt Not attempted
16/17 users correct (94%)

12pt Not attempted
9/15 users correct (60%)

Min Perimeter

5pt Not attempted
17/19 users correct (89%)

15pt Not attempted
4/13 users correct (31%)

Doubly-sorted Grid

10pt Not attempted
16/16 users correct (100%)

20pt Not attempted
4/5 users correct (80%)

Wi-fi Towers

3pt Not attempted
22/22 users correct (100%)

25pt Not attempted
9/12 users correct (75%)

Marbles

7pt Not attempted
16/19 users correct (84%)

32pt Not attempted
2/8 users correct (25%)

Lights

21pt Not attempted
2/4 users correct (50%)

45pt Not attempted
1/2 users correct (50%)

Top Scores

ACRush	168
qizichao	87
wata	81
ZhukovDmitry	70
dzhulgakov	69
nika	62
Vitaliy	62
kalinov	55
halyavin	54
bmerry	50

Problem B. Min Perimeter

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 5 points	Solve B-small
Large input 15 points	Solve B-large

Problem

You will be given a set of points with integer coordinates. You are asked to compute the smallest perimeter of a triangle with distinct vertexes from this set of points.

Input

When you are ready to submit a solution, the file you will download will contain Java source code that produces the input data. You should compile and run that program (see details at the bottom of the problem statement), and it will print the input data for the problem to the standard output stream. The program is guaranteed to run within 30 seconds for the small input, and 90 seconds for the large input if you don't have other resource-intensive processes running.

The first line of the input data gives you the number of cases, **T**. **T** test cases follow. Each test case contains on the first line the integer **n**, the number of points in the set. **n** lines follow, each line containing two integer numbers $\mathbf{x_i}$, $\mathbf{y_i}$. These are the coordinates of the i-th point. There may not be more than one point at the same coordinates.

Output

For each test case, output:

```
Case #X: Y
```

where **X** is the number of the test case and Y is the minimum perimeter. Answers with a relative or absolute error of at most 10^{-9} will be considered correct. Degenerate triangles — triangles with zero area — are ok.

Limits

1 <= **T** <= 15
$0 \le \mathbf{x_i}, \mathbf{y_i} \le 10^9$

Small dataset

3 <= **n** <= 10000

Large dataset

3 <= **n** <= 1000000

Sample

Input

```
public class Input {
  public static void main(String[] args) {
    // T = 1
    System.out.println(1);
    // n = 10
    System.out.println(10);
    // (x[i], y[i]) = (i, i)
    for (int i = 0; i < 10; i++)
      System.out.println("" + i + " " + i);
  }
}
```

Output

```
Case #1: 5.656854
```

How to compile and run downloaded Java program

Once you're ready to submit a solution, download the program, and run it using the following instructions to get the input file:

Linux instructions:

1. Rename the file to "Input.java" (without quotes).
2. Start a console shell (Applications -> Accessories -> Terminal) and set the current directory to the location of the file ("cd <directory name>").
3. Run

   ```
   /usr/lib/jvm/java-6-sun/bin/javac Input.java
   ```

   to compile the program.
4. Run

   ```
   /usr/lib/jvm/java-6-sun/bin/java -Xmx512M Input >test.txt
   ```

   to run the program and save the results to a file named "test.txt". Now that file contains the testcase that you need to solve.

Windows instructions:

1. Rename the file to "Input.java" (without quotes).
2. Start a console shell (Start->Run->cmd.exe and set the current directory to the location of the file ("cd <directory name>").
3. Run

   ```
   "C:\Program Files\Java\jdk1.6.0_16\bin\javac" Input.java
   ```

   to compile the program.
4. Run

   ```
   "C:\Program Files\Java\jdk1.6.0_16\bin\java" -Xmx512M Inpu
   ```

   to run the program and save the results to a file named "test.txt". Now that file contains the testcase that you need to solve.

World Finals 2009

A. Year of More Code Jam
B. Min Perimeter
C. Doubly-sorted Grid
D. Wi-fi Towers
E. Marbles
F. Lights

Contest Analysis
Questions asked 1

Submissions

Year of More Code Jam
5pt Not attempted **16/17 users** correct (94%)
12pt Not attempted **9/15 users** correct (60%)

Min Perimeter
5pt Not attempted **17/19 users** correct (89%)
15pt Not attempted **4/13 users** correct (31%)

Doubly-sorted Grid
10pt Not attempted **16/16 users** correct (100%)
20pt Not attempted **4/5 users** correct (80%)

Wi-fi Towers
3pt Not attempted **22/22 users** correct (100%)
25pt Not attempted **9/12 users** correct (75%)

Marbles
7pt Not attempted **16/19 users** correct (84%)
32pt Not attempted **2/8 users** correct (25%)

Lights
21pt Not attempted **2/4 users** correct (50%)
45pt Not attempted **1/2 users** correct (50%)

Top Scores

ACRush	168
qizichao	87
wata	81
ZhukovDmitry	70
dzhulgakov	69
nika	62
Vitaliy	62
kalinov	55
halyavin	54
bmerry	50

Problem C. **Doubly-sorted Grid**

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 20 points	Solve C-large

Problem

A rectangular grid with lower case English letters in each cell is called *doubly sorted* if in each row the letters are non-decreasing from the left to the right, and in each column the letters are non-decreasing from the top to the bottom. In the following examples, the first two grids are doubly sorted, while the other two are not:

```
abc    ace    aceg    base
def    ade    cdef    base
ghi    bdg    xxyy    base
```

You are given a partially-filled grid, where some of the cells are filled with letters. Your task is to compute the number of ways you can fill the rest of the cells so that the resulting grid is doubly sorted. The answer might be a big number; you need to output the number of ways modulo 10007.

Input

The first line of input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing two integers **R** and **C**, the number of rows and the number of columns respectively. This is followed by **R** lines, each containing a string of length **C**, giving the partially-filled grid. Each character in the grid is either a lower-case English letter, or '.', indicating that the cell is not filled yet.

Output

For each test case, output one line. That line should contain "Case #X: y", where **X** is the case number starting with 1, and **y** is the number of possible doubly-sorted grids, modulo 10007.

Limits

$1 \le \mathbf{T} \le 40$
Each character in the partially-filled grid is either '.' or a lower-case English letter.

Small dataset

$1 \le \mathbf{R}, \mathbf{C} \le 4$

Large dataset

$1 \le \mathbf{R}, \mathbf{C} \le 10$

Sample

Input	Output
3 2 2 ad c. 3 3 .a. a.z .z. 4 4 .... .g.. .cj.	Case #1: 23 Case #2: 7569 Case #3: 0

Problem D. Wi-fi Towers

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 3 points	Solve D-small
Large input 25 points	Solve D-large

Problem

You are given a network of wireless towers. Each tower has a range and can send data to neighboring towers as long as the distance is less than or equal to the sending tower's range.

The towers are using an old communication protocol A, but there is a new, better protocol B available. We are thinking about upgrading some towers to send data using protocol B to achieve better bandwidth.

There is one important restriction: if a tower T is using the new protocol B, every tower within T's range must also be running protocol B, so that they can understand the data sent from T. The reverse is not necessary — towers running the new protocol B can be sent data from towers using the old protocol A.

Your task is to select the best set of towers to upgrade from protocol A to protocol B. There is some benefit to upgrading a tower, but there are also installation costs. So each tower will have a score, which can be positive or negative, which is the value of upgrading the tower. Choose the set of towers to upgrade in such a way that the total score of the upgraded towers is maximized.

Input

The first line contains the number of test cases, **T**. Each test case starts with the number of towers, **n**. The following **n** lines each contain 4 integers: **x**, **y**, **r**, **s**. They describe a tower at coordinates **x**, **y** having a range of **r** and a score (value of updating to the new protocol) of **s**

Output

For each test case, output:

```
Case #X: score
```

where **X** is the test case number, starting from 1, and **score** is the total score for the best choice of towers.

Limits

$1 \leq \mathbf{T} \leq 55$
$-10\,000 \leq \mathbf{x}, \mathbf{y} \leq 10\,000$
$1 \leq \mathbf{r} \leq 20\,000$
$-1000 \leq \mathbf{s} \leq 1000$

No two towers will have the same coordinates.

Small dataset

$1 \leq \mathbf{n} \leq 15$

Large dataset

$1 \leq \mathbf{n} \leq 500$

Sample

Input	Output
1 5 0 1 7 10 0 -1 7 10 5 0 1 -15 10 0 6 10 15 1 2 -20	Case #1: 5

World Finals 2009

A. Year of More Code Jam

B. Min Perimeter

C. Doubly-sorted Grid

D. Wi-fi Towers

E. Marbles

F. Lights

Contest Analysis

Questions asked 1

Submissions

Year of More Code Jam

5pt Not attempted **16/17 users** correct (94%)

12pt Not attempted **9/15 users** correct (60%)

Min Perimeter

5pt Not attempted **17/19 users** correct (89%)

15pt Not attempted **4/13 users** correct (31%)

Doubly-sorted Grid

10pt Not attempted **16/16 users** correct (100%)

20pt Not attempted **4/5 users** correct (80%)

Wi-fi Towers

3pt Not attempted **22/22 users** correct (100%)

25pt Not attempted **9/12 users** correct (75%)

Marbles

7pt Not attempted **16/19 users** correct (84%)

32pt Not attempted **2/8 users** correct (25%)

Lights

21pt Not attempted **2/4 users** correct (50%)

45pt Not attempted **1/2 users** correct (50%)

Top Scores

ACRush	168
qizichao	87
wata	81
ZhukovDmitry	70
dzhulgakov	69
nika	62
Vitaliy	62
kalinov	55
halyavin	54
bmerry	50

Problem E. Marbles

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 7 points	Solve E-small
Large input 32 points	Solve E-large

Problem

You have 2**n** marbles on a square grid. The marbles are colored in **n** different colors such that there are exactly 2 marbles of each color. The marbles are placed at the coordinates (1,0), (2,0), ..., (2**n**, 0).

Your task is to draw a path for each color that joins the two marbles of that color. Each path should be composed of vertical or horizontal line segments between grid points. No two paths can intersect or touch each other. No path may cross the y=0 line. Each path can only touch the y=0 line at the position of the two marbles it is connecting, so the first and last line segment of each path must be vertical.

Given an arrangement of marbles, return the minimum height of a solution, or return -1 if no solution exists. The height is defined as the difference between the highest and lowest Y-coordinates of the paths used.

An example:

```
red red blue yellow blue yellow
```

One solution would be:

```
 +---+    +-----------+
 |   |    |           |
red red blue yellow blue yellow
                |           |
                +-----------+
```

The minimum height is 2 in this case.

Input

The first line of input gives the number of cases, **T**. **T** test cases follow. The first line of each case contains **n**, the number of different colors for the marbles. The next line contains a string of 2**n** words separated by spaces which correspond to the colors of the marbles, in order from left to right. Each color is a string of lower case letters ('a' .. 'z') no longer than 10 characters. There will be exactly **n** different colors and each color will appear exactly twice.

Output

For each test case, output one line containing "Case #**x**: ", where **x** is the case number (starting from 1), followed by the height of any optimal solution, or -1 if no solution exists.

Limits

1 <= **T** <= 50.

Small dataset

1 <= **n** <= 20.

Large dataset

1 <= **n** <= 500.

Sample

Input

```
4
3
red red blue yellow blue yellow
3
red blue yellow red blue yellow

3
red blue yellow blue yellow red
```

Output

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

```
3
red red blue blue yellow yellow
```

Problem F. Lights

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 21 points	Solve F-small
Large input 45 points	Solve F-large

Problem

In a big, square room there are two point light sources: one is red and the other is green. There are also **n** circular pillars.

Light travels in straight lines and is absorbed by walls and pillars. The pillars therefore cast shadows: they do not let light through. There are places in the room where no light reaches (black), where only one of the two light sources reaches (red or green), and places where both lights reach (yellow). Compute the total area of each of the four colors in the room. Do not include the area of the pillars.

Input

- One line containing the number of test cases, **T**.

Each test case contains, in order:

- One line containing the coordinates **x**, **y** of the red light source.
- One line containing the coordinates **x**, **y** of the green light source.
- One line containing the number of pillars **n**.
- **n** lines describing the pillars. Each contains 3 numbers **x**, **y**, **r**. The pillar is a disk with the center (**x**, **y**) and radius **r**.

The room is the square described by 0 ≤ **x**, **y** ≤ 100. Pillars, room walls and light sources are all disjoint, they do not overlap or touch.

Output

For each test case, output:

```
Case #X:
black area
red area
green area
yellow area
```

where **X** is the test case number, starting from 1, and each area is a real number.

Any answer with absolute or relative error of at most 10^{-5} will be accepted.

Limits

All input numbers are integers.

1 ≤ **T** ≤ 15
0 ≤ **x**, **y** ≤ 100
1 ≤ **r** ≤ 49

Small dataset

0 ≤ **n** ≤ 1

Large dataset

0 ≤ **n** ≤ 50

Sample

Input	Output
1 5 50 95 50 1 50 50 10	Case #1: 0.7656121 1437.986 1437.986 6809.104

Submissions

Year of More Code Jam
5pt Not attempted **16/17 users** correct (94%)
12pt Not attempted **9/15 users** correct (60%)

Min Perimeter
5pt Not attempted **17/19 users** correct (89%)
15pt Not attempted **4/13 users** correct (31%)

Doubly-sorted Grid
10pt Not attempted **16/16 users** correct (100%)
20pt Not attempted **4/5 users** correct (80%)

Wi-fi Towers
3pt Not attempted **22/22 users** correct (100%)
25pt Not attempted **9/12 users** correct (75%)

Marbles
7pt Not attempted **16/19 users** correct (84%)
32pt Not attempted **2/8 users** correct (25%)

Lights
21pt Not attempted **2/4 users** correct (50%)
45pt Not attempted **1/2 users** correct (50%)

Top Scores

ACRush	168
qizichao	87
wata	81
ZhukovDmitry	70
dzhulgakov	69
nika	62
Vitaliy	62
kalinov	55
halyavin	54
bmerry	50