Problem A. Country Leader

Problem

The Constitution of a certain country states that the leader is the person with the name containing the greatest number of different alphabet letters. (The country uses the uppercase English alphabet from A through Z.) For example, the name `GOOGLE` has four different alphabet letters: E, G, L, and O. The name `APAC CODE JAM` has eight different letters. If the country only consists of these 2 persons, `APAC CODE JAM` would be the leader.

If there is a tie, the person whose name comes earliest in alphabetical order is the leader.

Given a list of names of the citizens of the country, can you determine who the leader is?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line with an interger **N**, the number of people in the country. Then **N** lines follow. The i-th line represents the name of the i-th person. Each name contains at most 20 characters and contains at least one alphabet letter.

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 name of the leader.

Limits

1 ≤ **T** ≤ 100.
1 ≤ **N** ≤ 100.

Small dataset

Each name consists of at most 20 characters and only consists of the uppercase English letters A through Z.

Large dataset

Each name consists of at most 20 characters and only consists of the uppercase English letters A through Z and ' '(space).
All names start and end with alphabet letters.

Sample

Input	Output

`2`
`3`
`ADAM`
`BOB`
`JOHNSON`
`2`
`A AB C`
`DEF`
	`Case #1: JOHNSON`
	`Case #2: A AB C`

In sample case #1, `JOHNSON` contains 5 different alphabet letters('H', 'J', 'N', 'O', 'S'), so he is the leader.

Sample case #2 would only appear in Large data set. The name `DEF` contains 3 different alphabet letters, the name `A AB C` also contains 3 different alphabet letters. `A AB C` comes alphabetically earlier so he is the leader.

yashLadha	35
praran26	35
achaitanyasai	35
xhaler	35
iharsh234	35
Rajnikanth	35
sokokaleb	35
adtac	35
eon204	35
Irving.CL	35

Problem B. Vote

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 8 points — Solve B-large

Problem

A and B are the only two candidates competing in a certain election. We know from polls that exactly **N** voters support A, and exactly **M** voters support B. We also know that **N** is greater than **M**, so A will win.

Voters will show up at the polling place one at a time, in an order chosen uniformly at random from all possible (**N** + **M**)! orders. After each voter casts their vote, the polling place worker will update the results and note which candidate (if any) is winning so far. (If the votes are tied, neither candidate is considered to be winning.)

What is the probability that A stays in the lead the entire time -- that is, that A will always be winning after every vote?

Input

The input starts with one line containing one integer **T**, which is the number of test cases. Each test case consists of one line with two integers **N** and **M**: the numbers of voters supporting A and B, respectively.

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 probability that A will always be winning after every vote.

y will be considered correct if y is within an absolute or relative error of 10^{-6} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

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

Small dataset

$0 \le \mathbf{M} < \mathbf{N} \le 10$.

Large dataset

$0 \le \mathbf{M} < \mathbf{N} \le 2000$.

Sample

Input	Output
2 2 1 1 0	Case #1: 0.33333333 Case #2: 1.00000000

In sample case #1, there are 3 voters. Two of them support A -- we will call them A1 and A2 -- and one of them supports B. They can come to vote in six possible orders: A1 A2 B, A2 A1 B, A1 B A2, A2 B A1, B A1 A2, B A2 A1. Only the first two of those orders guarantee that Candidate A is winning after every vote. (For example, if the order is A1 B A2, then Candidate A is winning after the first vote but tied after the second vote.) So the answer is 2/6 = 0.333333...

In sample case #2, there is only 1 voter, and that voter supports A. There is only one possible order of arrival, and A will be winning after the one and only vote.

Kickstart Practice Round 2017

A. Country Leader

B. Vote

C. Sherlock and Parentheses

Questions asked 1

Submissions

Country Leader
4pt Not attempted **366/497 users** correct (74%)
7pt Not attempted **279/360 users** correct (78%)

Vote
5pt Not attempted **227/304 users** correct (75%)
8pt Not attempted **165/214 users** correct (77%)

Sherlock and Parentheses
4pt Not attempted **257/277 users** correct (93%)
7pt Not attempted **220/256 users** correct (86%)

Top Scores

yashLadha	35
praran26	35
achaitanyasai	35
xhaler	35
iharsh234	35
Rajnikanth	35
sokokaleb	35
adtac	35
eon204	35
Irving.CL	35

Problem C. Sherlock and Parentheses

Small input 4 points — Solve C-small

Large input 7 points — Solve C-large

Problem

Sherlock and Watson have recently enrolled in a computer programming course. Today, the tutor taught them about the balanced parentheses problem. A string S consisting only of characters (and/or) is *balanced* if:

- It is the empty string, or:
- It has the form (S), where S is a balanced string, or:
- It has the form S_1S_2, where S_1 is a balanced string and S_2 is a balanced string.

Sherlock coded up the solution very quickly and started bragging about how good he is, so Watson gave him a problem to test his knowledge. He asked Sherlock to generate a string S of **L + R** characters, in which there are a total of **L** left parentheses (and a total of **R** right parentheses). Moreover, the string must have as many different balanced non-empty substrings as possible. (Two substrings are considered different as long as they start or end at different indexes of the string, even if their content happens to be the same). Note that S itself does not have to be balanced.

Sherlock is sure that once he knows the maximum possible number of balanced non-empty substrings, he will be able to solve the problem. Can you help him find that maximum number?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with two integers: **L** and **R**.

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 answer, as described above.

Limits

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

Small dataset

$0 \le \mathbf{L} \le 20$.
$0 \le \mathbf{R} \le 20$.
$1 \le \mathbf{L} + \mathbf{R} \le 20$.

Large dataset

$0 \le \mathbf{L} \le 10^5$.
$0 \le \mathbf{R} \le 10^5$.
$1 \le \mathbf{L} + \mathbf{R} \le 10^5$.

Sample

Input	Output
3	Case #1: 0
1 0	Case #2: 1
1 1	Case #3: 3
3 2	

In Case 1, the only possible string is (. There are no balanced non-empty substrings.
In Case 2, the optimal string is (). There is only one balanced non-empty substring: the entire string itself.
In Case 3, both strings ()()(and (()() give the same optimal answer.
For the case ()()(, for example, the three balanced substrings are () from indexes 1 to 2, () from indexes 3 to 4, and ()() from indexes 1 to 4.

Kickstart Round A 2017

Questions asked 3

Submissions

Square Counting

8pt	Not attempted **1423/2010 users** correct (71%)
17pt	Not attempted **524/1333 users** correct (39%)

Patterns Overlap

13pt	Not attempted **394/1100 users** correct (36%)
22pt	Not attempted **287/364 users** correct (79%)

Space Cubes

14pt	Not attempted **252/395 users** correct (64%)
26pt	Not attempted **100/119 users** correct (84%)

Top Scores

Doju	100
phirasit	100
jerrymao	100
globalpointer	100
Kasugano.Sora	100
alecsyde	100
FatalEagle	100
xwchow	100
iskim	100
wifi	100

Problem A. Square Counting

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 17 points	Solve A-large

Problem

Mr. Panda has recently fallen in love with a new game called Square Off, in which players compete to find as many different squares as possible on an evenly spaced rectangular grid of dots. To find a square, a player must identify four dots that form the vertices of a square. Each side of the square must have the same length, of course, but it does not matter what that length is, and the square does not necessarily need to be aligned with the axes of the grid. The player earns one point for every different square found in this way. Two squares are different if and only if their sets of four dots are different.

Mr. Panda has just been given a grid with **R** rows and **C** columns of dots. How many different squares can he find in this grid? Since the number might be very large, please output the answer modulo $10^9 + 7$ (1000000007).

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each line has two integers **R** and **C**: the number of dots in each row and column of the grid, respectively.

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 different squares can be found in the grid.

Limits

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

Small dataset

$2 \le \mathbf{R} \le 1000$.
$2 \le \mathbf{C} \le 1000$.

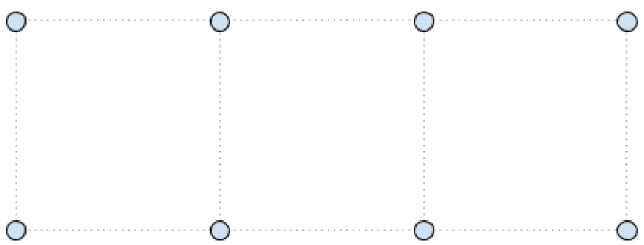
Large dataset

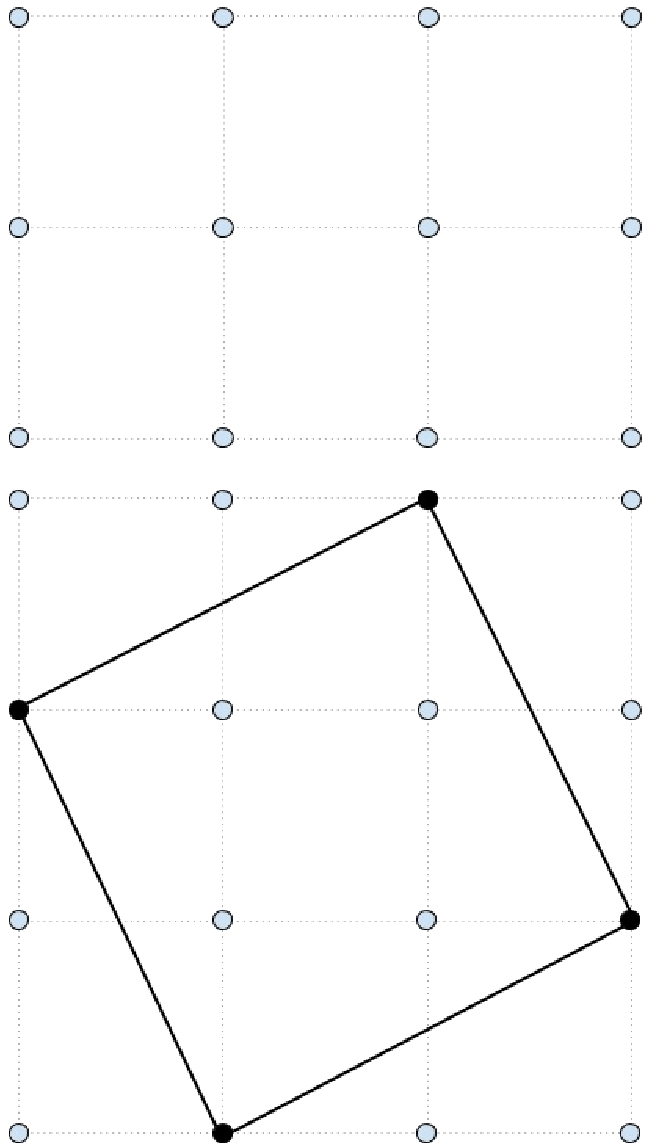
$2 \le \mathbf{R} \le 10^9$.
$2 \le \mathbf{C} \le 10^9$.

Sample

Input	Output
4 2 4 3 4 4 4 1000 500	Case #1: 3 Case #2: 10 Case #3: 20 Case #4: 624937395

The pictures below illustrate the grids from the three sample cases and a valid square in the third sample case.





Problem B. **Patterns Overlap**

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 22 points — Solve B-large

Problem

Alice likes reading and buys a lot of books. She stores her books in two boxes; each box is labeled with a pattern that matches the titles of all of the books stored in that box. A pattern consists of only uppercase/lowercase English alphabet letters and stars (*). A star can match between zero and four letters. For example, books with the titles `GoneGirl` and `GoneTomorrow` can be put in a box with the pattern `Gone**`, but books with the titles `TheGoneGirl`, and `GoneWithTheWind` cannot.

Alice is wondering whether there is any book that could be stored in either of the boxes. That is, she wonders if there is a title that matches both boxes' patterns.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each consists of two lines; each line has one string in which each character is either an uppercase/lowercase English letter or *.

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 TRUE if there is a string that matches both patterns, or FALSE if not.

Limits

1 ≤ **T** ≤ 50.

Small dataset

1 ≤ the length of each pattern ≤ 200.
Each pattern contains at most 5 stars.

Large dataset

1 ≤ the length of each pattern ≤ 2000.

Sample

Input	Output

`3`
`****`
`It`
`Shakes*e`
`S*speare`
`Shakes*e`
`*peare`
	`Case #1: TRUE`
	`Case #2: TRUE`
	`Case #3: FALSE`

In sample case #1, the title `It` matches both patterns. Note that it is possible for a * to match zero characters.

In sample case #2, the title `Shakespeare` matches both patterns.

In sample case #3, there is no title that matches both patterns. `Shakespeare`, for example, does not work because the * at the start of the `*peare` pattern cannot match six letters.

Problem C. Space Cubes

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 14 points — Solve C-small

Large input 26 points — Solve C-large

Problem

"Look at the stars, look how they shine for you." - Coldplay, "Yellow"

In a galaxy far, far away, there are many stars. Each one is a sphere with a certain position (in three-dimensional space) and radius. It is possible for stars to overlap each other.

The stars are so incredibly beautiful to you that you want to capture them forever! You would like to build two cubes of the same integer edge length, and place them in space such that for each star, there is at least one cube that *completely* contains it. (It's not enough for a star to be completely contained by the union of the two cubes.) A star is completely contained by a cube if no point on the star is outside the cube; a point exactly on a cube face is still considered to be inside the cube.

The cubes can be placed anywhere in space, but they must be placed with their edges parallel to the coordinate axes. It is acceptable for the cubes to overlap stars or each other.

What is the minimum integer edge length that allows you to achieve this goal?

Input

The input starts with one line containing exactly one integer **T**, which is the number of test cases. **T** test cases follow.

Each test case begins with a line containing an integer, **N**, representing the number of stars.

This is followed by **N** lines. On the ith line, there are 4 space-separated integers, $\mathbf{X_i}$, $\mathbf{Y_i}$, $\mathbf{Z_i}$ and $\mathbf{R_i}$, indicating the (X, Y, Z) coordinates of the center of the $\mathbf{i^{th}}$ star, and the radius of the $\mathbf{i^{th}}$ star.

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 cube edge length that solves the problem, as described above.

Limits

$1 \le \mathbf{T} \le 100$.
$-10^8 \le \mathbf{X_i} \le 10^8$, for all i.
$-10^8 \le \mathbf{Y_i} \le 10^8$, for all i.
$-10^8 \le \mathbf{Z_i} \le 10^8$, for all i.
$1 \le \mathbf{R_i} \le 10^8$, for all i.

Small dataset

$1 \le \mathbf{N} \le 16$.

Large dataset

$1 \le \mathbf{N} \le 2000$.

Sample

Input	Output
3	Case #1: 3
3	Case #2: 5
1 1 1 1	Case #3: 2
2 2 2 1	
4 4 4 1	
3	
1 1 1 2	
2 3 4 1	
5 6 7 1	
3	

```
1 1 1 1
1 1 1 1
9 9 9 1
```

In the first test case, one solution is to place two cubes with an edge length of 3 such that their corners with minimum (x, y, z) coordinates are at (0, 0, 0) and (3, 3, 3).
In the second test case, one solution is to place two cubes with an edge length of 5 such that their corners with minimum (x, y, z) coordinates are at (-1, -1, -1) and (1, 2, 3).

Problem A. Math Encoder

Small input
7 points
Solve A-small

Large input
16 points
Solve A-large

Problem

Professor Math is working on a secret project and is facing a challenge where a list of numbers need to be encoded into a single number in the most efficient manner. After much research, Professor Math finds a 3 step process that can best encode the numbers:

1. The first step is to find all possible non-empty subsets of the list of numbers and then, for each subset, find the difference between the largest and smallest numbers (that is, the largest minus the smallest) in that subset. Note that if there is only one number in a subset, it is both the largest and the smallest number in that subset. The complete set itself is also considered a subset.
2. Then add up all the differences to get the final encoded number.
3. As the number may be large, output the number modulo $10^9 + 7$ (1000000007).

The professor has shared an example and its explanation below. Given a list of numbers, can you help the professor build an efficient function to compute the final encoded number?

Input

The first line of the input gives the number of test cases, **T**. This is followed by T test cases where each test case is defined by 2 lines:

1. The first line gives a positive number **N**: the number of numbers in the list and
2. The second line contains a list of **N** positive integers $\mathbf{K_i}$, sorted in non-decreasing order.

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 final encoded number.

Since the output can be a really big number, we only ask you to output the remainder of dividing the result by the prime $10^9 + 7$ (1000000007).

Limits

$1 \le \mathbf{T} \le 25$.
$1 \le \mathbf{K_i} \le 10000$, for all i.
$\mathbf{K_i} \le \mathbf{K_{i+1}}$, for all i < **N** - 1.

Small dataset

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

Large dataset

$1 \le \mathbf{N} \le 10000$.

Sample

Input	Output
1 4 3 6 7 9	Case #1: 44

Explanation for the sample input

1. Find all subsets and get the difference between largest & smallest numbers:
[3], largest-smallest = 3 - 3 = 0.
[6], largest-smallest = 6 - 6 = 0.

[7], largest-smallest = 7 - 7 = 0.
[9], largest-smallest = 9 - 9 = 0.
[3, 6], largest-smallest = 6 - 3 = 3.
[3, 7], largest-smallest = 7 - 3 = 4.
[3, 9], largest-smallest = 9 - 3 = 6.
[6, 7], largest-smallest = 7 - 6 = 1.
[6, 9], largest-smallest = 9 - 6 = 3.
[7, 9], largest-smallest = 9 - 7 = 2.
[3, 6, 7], largest-smallest = 7 - 3 = 4.
[3, 6, 9], largest-smallest = 9 - 3 = 6.
[3, 7, 9], largest-smallest = 9 - 3 = 6.
[6, 7, 9], largest-smallest = 9 - 6 = 3.
[3, 6, 7, 9], largest-smallest = 9 - 3 = 6.

2. Find the sum of the differences calculated in the previous step:
   3+4+6+1+3+2+4+6+6+3+6
   = 44.
3. Find the answer modulo $10^9 + 7$ (1000000007):
   44 % 1000000007 = 44

Problem B. Center

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 21 points	Solve B-large

Problem

There are **N** weighted points in a plane. Point **i** is at ($\mathbf{X_i}$, $\mathbf{Y_i}$) and has weight $\mathbf{W_i}$.

In this problem, we need to find a special center of these points. The center is a point (**X**, **Y**) such that the sum of max(|**X**-$\mathbf{X_i}$|, |**Y**-$\mathbf{Y_i}$|)*$\mathbf{W_i}$ is minimum.

Input

The input starts with one line containing exactly one integer **T**, which is the number of test cases. **T** test cases follow.

Each test case begins with one line containing one integer **N**. **N** lines follow. Each line contains three space-separated real numbers $\mathbf{X_i}$, $\mathbf{Y_i}$, and $\mathbf{W_i}$. $\mathbf{X_i}$, $\mathbf{Y_i}$ and $\mathbf{W_i}$ have exactly 2 digits after the decimal point.

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 sum of max(|**X**-$\mathbf{X_i}$|, |**Y**-$\mathbf{Y_i}$|)*$\mathbf{W_i}$ for center (**X**, **Y**).

y will be considered correct if it is within an absolute or relative error of 10^{-6} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

1 ≤ **T** ≤ 10.
-1000.00 ≤ $\mathbf{X_i}$ ≤ 1000.00.
-1000.00 ≤ $\mathbf{Y_i}$ ≤ 1000.00.

Small dataset

1 ≤ **N** ≤ 100;
$\mathbf{W_i}$ = 1.0, for all i.

Large dataset

1 ≤ **N** ≤ 10000;
1.0 ≤ $\mathbf{W_i}$ ≤ 1000.0, for all i.

Sample

Input

```
3
2
0.00 0.00 1.00
1.00 0.00 1.00
4
1.00 1.00 1.00
1.00 -1.00 1.00
-1.00 1.00 1.00
-1.00 -1.00 1.00
2
0.00 0.00 1.00
1.00 0.00 2.00
```

Output

```
Case #1: 1.0
Case #2: 4.0
Case #3: 1.0
```

Problem C. Christmas 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 11 points	Solve C-small
Large input 32 points	Solve C-large

Problem

You are given a rectangular grid with **N** rows and **M** columns. Each cell of this grid is painted with one of two colors: green and white. Your task is to find the number of green cells in the largest Christmas tree in this grid.

To define a Christmas tree, first we define a **good triangle** as follows:

A good triangle with top point at row **R**, column **C** and height **h** (**h** > 0) is an isoceles triangle consisting entirely of green cells and pointing upward. Formally, this means that: The cell **(R, C)** is green, and for each **i** from 0 to **h-1** inclusive, the cells in row **R+i** from column **C-i** to column **C+i** are all green.

For example:

```
..#..
.####
#####
```
is a good triangle with height 3. The **#** cells are green and the **.** cells are white. Note that there is a green cell that is not part of the good triangle, even though it touches the good triangle.

```
..#..
.###.
####.
```
is **NOT** a good triangle with height 3, because the 5th cell in the 3rd row is white. However, there are good triangles with height 2 present.

```
...#.
.###.
#####.
```
is **NOT** a good triangle with height 3. However, there are good triangles with height 2 present.

A **K**-Christmas tree is defined as follows:

- It contains exactly **K** good triangles in vertical arrangement.
- The top cell of the **i+1**-th triangle must share its top edge with the bottom edge of any one of the cells at the base of the **i**-th triangle. This means that, if the base of the **i**-th triangle is at row **r**, from column **c1** to column **c2**, then the top of the **i+1**-th triangle must be on row **r+1**, in a column somewhere between **c1** and **c2**, inclusive.

For example, if **K** = 2:

```
...#...
..###..
.#####.
#######
..#....
.###...
#####..
```
is a valid 2-Christmas tree. Note that the height of the 2 good triangles can be different.

```
..#..
.###.
.#...
```
is also a valid 2-Christmas tree. Note that a good triangle can be of height 1 and have only one green cell.

```
...#...
..###..
.#####.
.......
..#....
.###...
#####..
```
The two good triangles with height 3 does **NOT** form a valid 2-Christmas tree, because the 2nd triangle must starts from the 4-th row.

```
...#.
..###
.#...
###..
```

The two good triangles with height 2 does **NOT** form a valid 2-Christmas tree, because the top of the 2nd triangle must be in a column between 3 and 5, inclusive.

You need to find the K-Christmas tree with the largest number of green cells.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of three lines:

- The first line contains 3 space-separated integers **N**, **M** and **K**, where **N** is the number of rows of the grid, **M** is the number of columns of the grid and **K** is the number of good triangle in the desired Christmas tree.
- The next **N** lines each contain exactly **M** characters. Each character will be either **.** or **#**, representing a white or green cell, respectively.

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 green cells in the largest K-Christmas tree. If there is no K-Christmas tree, output 0.

Limits

1 ≤ **T** ≤ 100.
1 ≤ **M** ≤ 100.
1 ≤ **N** ≤ 100.
Each cell in the grid is either `.` or #.

Small dataset

K = 1.

Large dataset

1 ≤ **K** ≤ 100.

Sample

```
Input      Output

4          Case #1: 9
3 5 1      Case #2: 0
..#..      Case #3: 9
.###.      Case #4: 10
#####
3 5 1
.....
.....
.....
4 5 1
#####
#####
#####
#####
4 5 2
#####
#####
#####
#####
```

In sample case #1, the largest 1-Christmas tree has 9 green cells:

```
..#..
.###.
#####
```

In sample case #2, there is no 1-Christmas tree.

In sample case #3, one largest 1-Christmas tree with 9 green cells is:

```
#####
#####
#####
#####
```

In sample case #4, one largest 2-Christmas tree with 10 green cells is:

```
#####
#####
#####
#####
```

Submissions

Diwali lightings

5pt Not attempted **89/141 users** correct (63%)

8pt Not attempted **62/87 users** correct (71%)

Safe Squares

6pt Not attempted **55/58 users** correct (95%)

13pt Not attempted **25/53 users** correct (47%)

Beautiful Numbers

6pt Not attempted **51/63 users** correct (81%)

15pt Not attempted **14/39 users** correct (36%)

Watson and Intervals

8pt Not attempted **12/15 users** correct (80%)

17pt Not attempted **7/10 users** correct (70%)

Top Scores

Benq	78
1717374	78
yubowenok	78
gridnevvvit	78
LiCode	65
Yash....	53
YourRatzon	53
broncos.billy	53
cmroz	53
sam1373	50

Problem A. Diwali lightings

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 8 points — Solve A-large

Problem

Diwali is the festival of lights. To celebrate it, people decorate their houses with multi-color lights and burst crackers. Everyone loves Diwali, and so does Pari. Pari is very fond of lights, and has transfinite powers, so she buys an infinite number of red and blue light bulbs. As a programmer, she also loves patterns, so she arranges her lights by infinitely repeating a given finite pattern **S**.

For example, if **S** is BBRB, the infinite sequence Pari builds would be BBRBBBRBBBRB...

Blue is Pari's favorite color, so she wants to know the number of blue bulbs between the **I**th bulb and **J**th bulb, inclusive, in the infinite sequence she built (lights are numbered with consecutive integers starting from 1). In the sequence above, the indices would be numbered as follows:

```
B  B  R  B  B  B  R  B  B  B  R  B...
1  2  3  4  5  6  7  8  9  10 11 12
```

So, for example, there are 4 blue lights between the 4th and 8th positions, but only 2 between the 10th and 12th.

Since the sequence can be very long, she wrote a program to do the count for her. Can you do the same?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.
First line of each test case consists of a string S, denoting the initial finite pattern.
Second line of each test case consists of two space separated integers **I** and **J**, defined above.

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 number of blue bulbs between the **I**th bulb and **J**th bulb of Pari's infinite sequence, inclusive.

Limits

1 ≤ **T** ≤ 100.
1 ≤ length of **S** ≤ 100.
Each character of **S** is either uppercase B or uppercase R.

Small dataset

1 ≤ **I** ≤ **J** ≤ 10^6.

Large dataset

1 ≤ **I** ≤ **J** ≤ 10^{18}.

Sample

Input	Output
3 BBRB 4 8 BBRB 10 12 BR 1 1000000	Case #1: 4 Case #2: 2 Case #3: 500000

Cases #1 and #2 are explained above.

In Case #3, bulbs at odd indices are always blue, and bulbs at even indices are always red, so there are half a million blue bulbs between positions 1 and 10^6.

Problem B. Safe Squares

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

Codejamon trainers are actively looking for monsters, but if you are not a trainer, these monsters could be really dangerous for you. You might want to find safe places that do not have any monsters!

Consider our world as a grid, and some of the cells have been occupied by monsters. We define a *safe square* as a grid-aligned **D** × **D** square of grid cells (with **D** ≥ 1) that does not contain any monsters. Your task is to find out how many safe squares (of any size) we have in the entire world.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line with three integers, **R**, **C**, and **K**. The grid has **R** rows and **C** columns, and contains **K** monsters. **K** more lines follow; each contains two integers $\mathbf{R_i}$ and $\mathbf{C_i}$, indicating the row and column that the i-th monster is in. (Rows are numbered from top to bottom, starting from 0; columns are numbered from left to right, starting from 0.)

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 safe zones for this test case.

Limits

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

$(\mathbf{R_i}, \mathbf{C_i}) \ne (\mathbf{R_j}, \mathbf{C_j})$ for i ≠ j. (No two monsters are in the same grid cell.)

$0 \le \mathbf{R_i} < \mathbf{R}$, i from 1 to **K**

$0 \le \mathbf{C_i} < \mathbf{C}$, i from 1 to **K**

Small dataset

$1 \le \mathbf{R} \le 10$.

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

$0 \le \mathbf{K} \le 10$.

Large dataset

$1 \le \mathbf{R} \le 3000$.

$1 \le \mathbf{C} \le 3000$.

$0 \le \mathbf{K} \le 3000$.

Sample

Input	Output
2 3 3 1 2 1 4 11 12 0 1 0 3 0 4 0 10 1 0 1 9 2 0 2 4 2 9 2 10 3 4 3 10	Case #1: 10 Case #2: 51

The grid of sample case #1 is:

```
0 0 0
0 0 0
0 1 0
```

Here, 0 represents a cell with no monster, and 1 represents a cell with a monster. It has 10 safe squares: 8 1x1 and 2 2x2.

The grid of sample case #2 is:

```
0 1 0 1 1 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 1 0
1 0 0 0 1 0 0 0 0 1 1
0 0 0 0 1 0 0 0 0 0 1
```

Note that sample case #2 will only appear in the Large dataset. It has 51 safe squares: 32 1x1, 13 2x2, 5 3x3, and 1 4x4.

Submissions

Diwali lightings

5pt Not attempted **89/141 users** correct (63%)

8pt Not attempted **62/87 users** correct (71%)

Safe Squares

6pt Not attempted **55/58 users** correct (95%)

13pt Not attempted **25/53 users** correct (47%)

Beautiful Numbers

6pt Not attempted **51/63 users** correct (81%)

15pt Not attempted **14/39 users** correct (36%)

Watson and Intervals

8pt Not attempted **12/15 users** correct (80%)

17pt Not attempted **7/10 users** correct (70%)

Top Scores

Benq	78
1717374	78
yubowenok	78
gridnevvvit	78
LiCode	65
Yash....	53
YourRatzon	53
broncos.billy	53
cmroz	53
sam1373	50

Problem C. Beautiful Numbers

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 C-small
Large input 15 points	Solve C-large

Problem

We consider a number to be *beautiful* if it consists only of the digit 1 repeated one or more times. Not all numbers are beautiful, but we can make any base 10 positive integer beautiful by writing it in another base.

Given an integer **N**, can you find a base B (with $B > 1$) to write it in such that all of its digits become 1? If there are multiple bases that satisfy this property, choose the one that maximizes the number of 1 digits.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with an integer **N**.

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 base described in the problem statement.

Limits

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

Small dataset

$3 \le \mathbf{N} \le 1000$.

Large dataset

$3 \le \mathbf{N} \le 10^{18}$.

Sample

Input	Output
2 3 13	Case #1: 2 Case #2: 3

In case #1, the optimal solution is to write 3 as 11 in base 2.

In case #2, the optimal solution is to write 13 as 111 in base 3. Note that we could also write 13 as 11 in base 12, but neither of those representations has as many 1s.

Problem D. **Watson and Intervals**

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 D-small
Large input 17 points	Solve D-large

Problem

Sherlock and Watson have mastered the intricacies of the language C++ in their programming course, so they have moved on to algorithmic problems. In today's class, the tutor introduced the problem of merging one-dimensional intervals. **N** intervals are given, and the ith interval is defined by the inclusive endpoints [$\mathbf{L_i}$, $\mathbf{R_i}$], where $\mathbf{L_i} \le \mathbf{R_i}$.

The tutor defined the *covered area* of a set of intervals as the number of integers appearing in at least one of the intervals. (Formally, an integer p contributes to the covered area if there is some j such that $\mathbf{L_j} \le p \le \mathbf{R_j}$.)

Now, Watson always likes to challenge Sherlock. He has asked Sherlock to remove exactly one interval such that the covered area of the remaining intervals is minimized. Help Sherlock find this minimum possible covered area, after removing exactly one of the **N** intervals.

Input

Each test case consists of one line with eight integers **N**, $\mathbf{L_1}$, $\mathbf{R_1}$, **A**, **B**, $\mathbf{C_1}$, $\mathbf{C_2}$, and **M**. **N** is the number of intervals, and the other seven values are parameters that you should use to generate the other intervals, as follows:

First define $x_1 = \mathbf{L_1}$ and $y_1 = \mathbf{R_1}$. Then, use the recurrences below to generate x_i, y_i for i = 2 to **N**:

- $x_i = (\mathbf{A}*x_{i-1} + \mathbf{B}*y_{i-1} + \mathbf{C_1})$ modulo **M**.
- $y_i = (\mathbf{A}*y_{i-1} + \mathbf{B}*x_{i-1} + \mathbf{C_2})$ modulo **M**.

We define $\mathbf{L_i} = \min(x_i, y_i)$ and $\mathbf{R_i} = \max(x_i, y_i)$, for all i = 2 to **N**.

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 possible covered area of all of the intervals remaining after removing exactly one interval.

Limits

$1 \le \mathbf{T} \le 50$.
$0 \le \mathbf{L_1} \le \mathbf{R_1} \le 10^9$.
$0 \le \mathbf{A} \le 10^9$.
$0 \le \mathbf{B} \le 10^9$.
$0 \le \mathbf{C_1} \le 10^9$.
$0 \le \mathbf{C_2} \le 10^9$.
$1 \le \mathbf{M} \le 10^9$.

Small dataset

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

Large dataset

$1 \le \mathbf{N} \le 5 * 10^5$(500000).

Sample

Input	Output
3 1 1 1 1 1 1 1 1 3 2 5 1 2 3 4 10 4 3 4 3 3 8 10 10	Case #1: 0 Case #2: 4 Case #3: 9

In case 1, using the generation method, the set of intervals generated are: {[1, 1]}. Removing the only interval, the *covered area* is 0.

In case 2, using the generation method, the set of intervals generated are: {[2, 5], [3, 5], [4, 7]}. Removing the first, second or third interval would cause the covered area of remaining intervals to be 5, 6 and 4, respectively.

In case 3, using the generation method, the set of intervals generated are: {[3, 4], [1, 9], [0, 8], [2, 4]}. Removing the first, second, third or fourth interval would cause the covered area of remaining intervals to be 10, 9, 9 and 10, respectively.

Submissions

Ambiguous Cipher

7pt Not attempted **813/966 users** correct (84%)

12pt Not attempted **683/755 users** correct (90%)

X Squared

9pt Not attempted **377/706 users** correct (53%)

14pt Not attempted **319/358 users** correct (89%)

Magical Thinking

6pt Not attempted **570/621 users** correct (92%)

19pt Not attempted **149/325 users** correct (46%)

The 4M Corporation

11pt Not attempted **109/194 users** correct (56%)

22pt Not attempted **60/78 users** correct (77%)

Top Scores

ACMonster	100
subscriber	100
Kasugano.Sora	100
spnautilus	100
1717374	100
Benq	100
LeeSin	100
yubowenok	100
praran26	100
cephian	100

Problem A. Ambiguous Cipher

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 12 points	Solve A-large

Problem

Susie and Calvin are classmates. Calvin would like to be able to pass notes to Susie in class without their teacher or other classmates knowing what they are talking about, just in case the notes fall into the wrong hands. Calvin has devised the a system to encrypt his messages.

Calvin only passes one word to Susie each time, and that word consists of only uppercase letters, because Calvin is so excited to talk to Susie. Each word is encrypted as follows:

- Calvin assigns a number to each letter based on the letter's position in the alphabet, where A = 0, B = 1, ..., Z = 25.
- For every letter in the word, Calvin determines the encrypted value of the letter by summing the values of the 1 or 2 letter(s) that are adjacent to that letter in the word. He takes that sum modulo 26, and this is the new value of the letter. Calvin then converts the value back to an uppercase letter based on positions in the alphabet, as before.
- The encrypted word is determined by encrypting every letter in the word using this method. Each letter's encryption is based only on the letters from the original unencrypted message, and not on any letters that have already been encrypted

Let's take a look at one of the notes Calvin is writing for Susie. Since Calvin is always hungry, he wants to let Susie know that he wants to eat again. Calvin encrypts the word `SOUP` as follows:

- S = 18, O = 14, U = 20, and P = 15.
- Calvin encrypts each letter based on the values of its neighbor(s):
  - First letter: 14 mod 26 = 14.
  - Second letter: (18 + 20) mod 26 = 12.
  - Third letter: (14 + 15) mod 26 = 3.
  - Fourth letter: 20 mod 26 = 20.
- The values 14 12 3 20 correspond to the letters `OMDU`, and this is the encrypted word that Calvin will write on the note for Susie.

It is guaranteed that Calvin will not send Susie any words that cannot be decrypted at all. For example, Calvin would not send Susie the word `APE`, since it does not have any valid decryptions. (That is, there is no word that Calvin could have encrypted to `APE`.)

However, Calvin's system is not perfect, and some of the words he sends Susie can actually be decrypted to multiple words, creating ambiguity! For example, `BCB` can be decrypted to `ABC` or `CBA`, among other possibilities.

Susie pulled another all-nighter yesterday to finish school projects, and she is too tired to decrypt Calvin's messages. She needs your help!

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each case is a single line that contains a string **W** of uppercase letters: an encrypted word that Calvin has sent.

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 decrypted word, or `AMBIGUOUS` if it is impossible to uniquely determine the decrypted word.

Limits

$1 \leq \mathbf{T} \leq 100$.
W consists of only of uppercase English letters.
W is decryptable to one or more words. (That is, **W** is the result of an encryption of some word.)
W does not decrypt to the word `AMBIGUOUS`. (You will only output that when the decryption is ambiguous.)

Small dataset

$2 \leq$ the length of $\mathbf{W} \leq 4$.

Large dataset

2 ≤ the length of **W** ≤ 50.

Sample

Input	Output
3 OMDU BCB AOAAAN	Case #1: SOUP Case #2: AMBIGUOUS Case #3: BANANA

Note that the last sample case would not appear in the Small dataset.

Sample Cases #1 & #2 were explained in the problem statement.

In Sample Case #3, `BANANA` is the only word that encrypts to `AOAAAN`.

Submissions

Ambiguous Cipher

7pt Not attempted **813/966 users** correct (84%)

12pt Not attempted **683/755 users** correct (90%)

X Squared

9pt Not attempted **377/706 users** correct (53%)

14pt Not attempted **319/358 users** correct (89%)

Magical Thinking

6pt Not attempted **570/621 users** correct (92%)

19pt Not attempted **149/325 users** correct (46%)

The 4M Corporation

11pt Not attempted **109/194 users** correct (56%)

22pt Not attempted **60/78 users** correct (77%)

Top Scores

ACMonster	100
subscriber	100
Kasugano.Sora	100
spnautilus	100
1717374	100
Benq	100
LeeSin	100
yubowenok	100
praran26	100
cephian	100

Problem B. X Squared

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 14 points	Solve B-large

Problem

The hot new toy for this year is called "X Squared". It consists of a square **N** by **N** grid of tiles, where **N** is odd. Exactly 2 × **N** - 1 of the tiles are labeled with an X, and the rest are blank (which we will represent with the `.` character). In each move of the game, the player can either choose and exchange two rows of tiles, or choose and exchange two columns of tiles. The goal of the game is to get all of the X tiles to be on the two main diagonals of the grid, forming a larger X shape, as in the following example for **N** = 5:

```
X...X
.X.X.
..X..
.X.X.
X...X
```

You are about to play with your X Squared toy, which is not yet in the goal state. You suspect that your devious younger sibling might have moved some of the tiles around in a way that has broken the game. Given the current configuration of the grid, can you determine whether it is possible to win or not?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each one begins with one line with an integer **N**, the size of the grid. **N** more lines with **N** characters each follow; the j-th character on the i-th of these lines is X if the tile in the i-th row and j-th column of the grid has an X, or `.` if that tile is blank.

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 `POSSIBLE` if it is possible to win, and `IMPOSSIBLE` otherwise.

Limits

1 ≤ **T** ≤ 100.
N mod 2 = 1. (**N** is odd.)
The grid has exactly 2 × **N** - 1 X tiles and exactly $\mathbf{N}^2$ - 2 × **N** + 1 `.` tiles.
The grid is not already in the goal state, as described in the problem statement.

Small dataset

3 ≤ **N** ≤ 5.

Large dataset

3 ≤ **N** ≤ 55.

Sample

Input	Output
2 3 ..X XX. XX. 3 ... XXX XX.	Case #1: POSSIBLE Case #2: IMPOSSIBLE

In Sample Case #1, one winning strategy is:

1. Swap the top row with the middle row.
2. Swap the rightmost column with the middle column.

```
..X    XX.    X.X
XX. -> ..X -> .X.
XX.    XX.    X.X
```

In Sample Case #2, no sequence of moves can turn the grid into the desired final configuration.

Problem C. Magical Thinking

Problem

You and **N** of your friends just took the B.A.T. (Binary Answer Test) to try to get into wizard school. The B.A.T. has **Q** true-false questions, and each one is worth 1 point. You have no wizard powers, so you just picked arbitrary answers and hoped for the best.

The results of the test have already been sent out by quail mail, but the quail with your results has not arrived yet. However, each of your friends has told you their list of answers and their total score. You also remember your own list of answers. You are an optimist and you think that you probably did well!

Given that there is one correct list of answers (but you do not know what those answers are), and given your friends' answers and scores, what is the highest score that you possibly could have achieved?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each begins with one line with two integers **N** and **Q**. Then, **N**+1 lines follow; the i-th of these lines represents the i-th examinee's list of answers $\mathbf{A_i}$, and has **Q** characters, each of which is either T or F (representing True or False). $\mathbf{A_{N+1}}$ is your own list of answers. Finally, one line with **N** integers follows; the i-th of these integers, $\mathbf{S_i}$, represents the i-th examinee's score. (Note that your own score is not in this list, because it is unknown.)

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 highest score that you possibly could have achieved that is consistent with the given information.

Limits

$1 \le \mathbf{T} \le 100$.
The length of $\mathbf{A_i}$ is **Q**, for all i.
Each character of $\mathbf{A_i}$ is either T or F, for all i.
$0 \le \mathbf{S_i} \le \mathbf{Q}$.
It is guaranteed that there is at least one possible list of correct answers that is consistent with all of the friends' answers and scores.

Small dataset

$\mathbf{N} = 1$.
$1 \le \mathbf{Q} \le 10$.

Large dataset

$1 \le \mathbf{N} \le 2$.
$1 \le \mathbf{Q} \le 50$.

Sample

Input	Output
3 1 2 TF FF 1 1 3 TTT TTF 0 2 3 TTF FTF TTT 1 2	Case #1: 2 Case #2: 1 Case #3: 2

Note that the last sample case would not appear in the Small dataset.

In sample case #1, your friend answered TF and you answered FF, and exactly one of your friend's answers was right. If your friend was wrong on question 1 and right on question 2, then the real set of answers is FF and you got both questions right. It is impossible to do better than this!

In sample case #2, your friend answered all Ts and got all of the questions wrong, so the real set of answers must be all Fs, which means that you got only question 3 right.

In sample case #3, the only possible real lists of answers that are consistent with the given information are FTT and FFF. (For example, the real answer list cannot be TFT; the first friend's answers and score would be consistent with that, but the second friend would have scored 0 instead of 2.) Of these two possibilities, FTT is more favorable to you and would give you a score of 2.

Problem D. The 4M Corporation

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
22 points
Solve D-large

Problem

The 4M Corporation has hired you to organize their departments and allocate headcount. You will create at least one department, and each department will receive some positive integer number of employees. It will not be easy, though — you have four different bosses, and each has given you a different instruction:

1. The department with the fewest employees must have exactly **MINIMUM** employees.
2. The department with the most employees must have exactly **MAXIMUM** employees.
3. The average number of employees across all departments must be exactly **MEAN**.
4. The median of the number of employees across all departments must be exactly **MEDIAN**. As a reminder, the median of a list is the value that, when the list is sorted in nondecreasing order, is in the center (for a list of odd length) or is the average of the two values in the center (for a list of even length).

Moreover, for the sake of efficiency, it is best to avoid creating too many departments. What is the smallest number of departments that you can create, if it is possible to satisfy your bosses' requests?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each consists of four integers: **MINIMUM**, **MAXIMUM**, **MEAN**, and **MEDIAN**, in that order.

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 either the minimum possible number of departments, or `IMPOSSIBLE` if it is impossible to satisfy all four bosses' requests.

Limits

1 ≤ **T** ≤ 100.

Small dataset

1 ≤ **MINIMUM** ≤ 8.
1 ≤ **MAXIMUM** ≤ 8.
1 ≤ **MEAN** ≤ 8.
1 ≤ **MEDIAN** ≤ 8.
The constraints for the Small dataset guarantee that the answer is either `IMPOSSIBLE` or is less than 14.

Large dataset

1 ≤ **MINIMUM** ≤ 10000.
1 ≤ **MAXIMUM** ≤ 10000.
1 ≤ **MEAN** ≤ 10000.
1 ≤ **MEDIAN** ≤ 10000.

Sample

Input	Output
5	
6 4 5 1	Case #1: IMPOSSIBLE
7 7 8 8	Case #2: IMPOSSIBLE
2 2 2 2	Case #3: 1
3 7 5 5	Case #4: 2
1 4 3 4	Case #5: 3

Sample Case #1 is `IMPOSSIBLE` because the maximum value cannot be smaller than the minimum value.

Sample Case #2 is `IMPOSSIBLE` because the mean and median cannot be larger than the maximum value.

In Sample Case #3, you can create a single department with 2 employees. This satisfies all four bosses: the department with the fewest employees has exactly 2, the department with the most employees has exactly 2, and the mean and median are both 2.

In Sample Case #4, you can create one department with 3 employees and another department with 7 employees. Note that it would **not** suffice to create only one department with 5 employees, because then the department with the fewest employees would not have exactly 3 and the department with the most employees would not have exactly 7.

For Sample Case #5, you can create one department with 1 employee and two more departments with 4 employees each.

Problem A. Go Sightseeing

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 14 points	Solve A-large

Problem

When you travel, you like to spend time sightseeing in as many cities as possible, but sometimes you might not be able to because you need to catch the bus to the next city. To maximize your travel enjoyment, you decide to write a program to optimize your schedule.

You begin at city 1 at time 0 and plan to travel to cities 2 to **N** in ascending order, visiting every city. There is a bus service from every city i to the next city i + 1. The i-th bus service runs on a schedule that is specified by 3 integers: $\mathbf{S_i}$, $\mathbf{F_i}$ and $\mathbf{D_i}$, the start time, frequency and ride duration. Formally, this means that there is a bus leaving from city i at all times $\mathbf{S_i} + x\mathbf{F_i}$, where x is an integer and x ≥ 0, and the bus takes $\mathbf{D_i}$ time to reach city i + 1.

At each city between 1 and **N** - 1, inclusive, you can decide to spend $\mathbf{T_s}$ time sightseeing before waiting for the next bus, or you can immediately wait for the next bus. You cannot go sightseeing multiple times in the same city. You may assume that boarding and leaving buses takes no time. You must arrive at city **N** by time $\mathbf{T_f}$ at the latest. (Note that you cannot go sightseeing in city **N**, even if you arrive early. There's nothing to see there!)

What is the maximum number of cities you can go sightseeing in?

Input

The input starts with one line containing one integer **T**, which is the number of test cases. **T** test cases follow.

Each test case begins with a line containing 3 integers, **N**, $\mathbf{T_s}$ and $\mathbf{T_f}$, representing the number of cities, the time taken for sightseeing in any city, and the latest time you can arrive in city **N**.

This is followed by **N** - 1 lines. On the i-th line, there are 3 integers, $\mathbf{S_i}$, $\mathbf{F_i}$ and $\mathbf{D_i}$, indicating the start time, frequency, and duration of buses travelling from city i to city i + 1.

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 maximum number of cities you can go sightseeing in such that you can still arrive at city **N** by time $\mathbf{T_f}$ at the latest. If it is impossible to arrive at city **N** by time $\mathbf{T_f}$, output `Case #x: IMPOSSIBLE`.

Limits

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

Small dataset

$2 \le \mathbf{N} \le 16$.
$1 \le \mathbf{S_i} \le 5000$.
$1 \le \mathbf{F_i} \le 5000$.
$1 \le \mathbf{D_i} \le 5000$.
$1 \le \mathbf{T_s} \le 5000$.
$1 \le \mathbf{T_f} \le 5000$.

Large dataset

$2 \le \mathbf{N} \le 2000$.
$1 \le \mathbf{S_i} \le 10^9$.
$1 \le \mathbf{F_i} \le 10^9$.
$1 \le \mathbf{D_i} \le 10^9$.
$1 \le \mathbf{T_s} \le 10^9$.
$1 \le \mathbf{T_f} \le 10^9$.

Sample

Input	Output
4	Case #1: 2
4 3 12	Case #2: 0
3 2 1	Case #3: IMPOSSIBLE
6 2 2	Case #4: 4
1 3 2	
3 2 30	
1 2 27	
3 2 1	
4 1 11	
2 1 2	
4 1 5	
8 2 2	
5 10 5000	
14 27 31	
27 11 44	
30 8 20	
2000 4000 3	

In the first test case, you can go sightseeing in city 1, catching the bus leaving at time 3 and arriving at time 4. You can go sightseeing in city 2, leaving on the bus at time 8. When you arrive in city 3 at time 10 you immediately board the next bus and arrive in city 4 just in time at time 12.

Problem B. Sherlock and The Matrix 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 13 points — Solve B-small

Large input 19 points — Solve B-large

Problem

Today, Sherlock and Watson attended a lecture in which they were introduced to matrices. Sherlock is one of those programmers who is not really interested in linear algebra, but he did come up with a problem involving matrices for Watson to solve.

Sherlock has given Watson two one-dimensional arrays A and B; both have length **N**. He has asked Watson to form a matrix with **N** rows and **N** columns, in which the j^{th} element in the i^{th} row is the product of the i-th element of A and the j-th element of B.

Let (x, y) denote the cell of the matrix in the x-th row (numbered starting from 0, starting from the top row) and the y-th column (numbered starting from 0, starting from the left column). Then a submatrix is defined by bottom-left and top-right cells `(a, b)` and `(c, d)` respectively, with a ≥ c and d ≥ b, and the submatrix consists of all cells (i, j) such that c ≤ i ≤ a and b ≤ j ≤ d. The sum of a submatrix is defined as sum of all of the cells of the submatrix.

To challenge Watson, Sherlock has given him an integer **K** and asked him to output the $\mathbf{K}^{th}$ largest sum among all submatrices in Watson's matrix, with **K** counting starting from 1 for the largest sum. (It is possible that different values of **K** may correspond to the same sum; that is, there may be multiple submatrices with the same sum.) Can you help Watson?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with nine integers **N**, **K**, $\mathbf{A_1}$, $\mathbf{B_1}$, **C**, **D**, $\mathbf{E_1}$, $\mathbf{E_2}$ and **F**. **N** is the length of arrays A and B; **K** is the ranking of the submatrix sum Watson has to output, $\mathbf{A_1}$ and $\mathbf{B_1}$ are the first elements of arrays A and B, respectively; and the other five values are parameters that you should use to generate the elements of the arrays, as follows:

First define $x_1 = \mathbf{A_1}$, $y_1 = \mathbf{B_1}$, $r_1 = 0$, $s_1 = 0$. Then, use the recurrences below to generate x_i and y_i for i = 2 to **N**:

- $x_i = (\mathbf{C}*x_{i-1} + \mathbf{D}*y_{i-1} + \mathbf{E_1})$ modulo **F**.
- $y_i = (\mathbf{D}*x_{i-1} + \mathbf{C}*y_{i-1} + \mathbf{E_2})$ modulo **F**.

Further, generate r_i and s_i for i = 2 to **N** using following recurrences:

- $r_i = (\mathbf{C}*r_{i-1} + \mathbf{D}*s_{i-1} + \mathbf{E_1})$ modulo 2.
- $s_i = (\mathbf{D}*r_{i-1} + \mathbf{C}*s_{i-1} + \mathbf{E_2})$ modulo 2.

We define $\mathbf{A_i} = (-1)^{r_i} * x_i$ and $\mathbf{B_i} = (-1)^{s_i} * y_i$, for all i = 2 to **N**.

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 $\mathbf{K}^{th}$ largest submatrix sum in the matrix defined in the statement.

Limits

$1 \le \mathbf{T} \le 20$.
$1 \le \mathbf{K} \le \min(10^5$, total number of submatrices possible).
$0 \le \mathbf{A_1} \le 10^3$.
$0 \le \mathbf{B_1} \le 10^3$.
$0 \le \mathbf{C} \le 10^3$.
$0 \le \mathbf{D} \le 10^3$.
$0 \le \mathbf{E_1} \le 10^3$.
$0 \le \mathbf{E_2} \le 10^3$.
$1 \le \mathbf{F} \le 10^3$.

Small dataset

$1 \le \mathbf{N} \le 200$.

Large dataset

$1 \leq \mathbf{N} \leq 10^5$.

Sample

Input	Output
4	Case #1: 6
2 3 1 1 1 1 1 1 5	Case #2: 4
1 1 2 2 2 2 2 2 5	Case #3: 1
2 3 1 2 2 1 1 1 5	Case #4: 42
9 8 7 6 5 4 3 2 1	

In case 1, using the generation method, the generated arrays A and B are [1, -3] and [1, -3], respectively. So, the matrix formed is
[1, -3]
[-3, 9]
All possible submatrix sums in decreasing order are [9, 6, 6, 4, 1, -2, -2, -3, -3].
As **K = 3**, answer is 6.

In case 2, using the generation method, the generated arrays A and B are [2] and [2], respectively. So, the matrix formed is
[4]
As **K = 1**, answer is 4.

In case 3, using the generation method, the generated arrays A and B are [1, 0] and [2, -1] respectively. So, the matrix formed is
[2, -1]
[0, 0]
All possible submatrix sums in decreasing order are [2, 2, 1, 1, 0, 0, 0, -1, -1].
As **K = 3**, answer is 1.

Problem C. Trash Throwing

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 17 points	Solve C-small
Large input 27 points	Solve C-large

Problem

Bob is an outstanding Googler. He loves efficiency, so he does everything well and quickly. Today, Bob has discovered that the trash can near his desk has disappeared! Sadly, this means that he has to use another nearby trash can instead. Since getting out of his seat to use the trash can would lower his productivity, Bob has decided to *throw* his trash into that trash can!

But there are many obstacles in the Google office. For example, it is rude if the thrown trash hits somebody, or the wall, or anything else. Bob hopes to throw the trash without touching any existing obstacles.

To simplify this problem, we will only consider the vertical plane that includes Bob and the trash can. Bob is at point (0, 0); the trash can is at point (**P**, 0). Moreover, there are **N** obstacles in the office; each of them is a single point, and the i-th one has coordinates ($\mathbf{X_i}$, $\mathbf{Y_i}$). The ceiling of the office is a line with the expression **y=H** in the plane. Since Bob is in one of the new high-tech floating offices, we do not consider the office floor in this problem; you do not need to worry about collisions with it. Bob will throw a piece of trash that is a circle with radius *R*. The center of the piece of trash starts off at (0, 0). When the piece of trash is thrown, the center of the piece of trash must follow the path of a parabola with the expression f(x)=*a*x(x-**P**), where 0 ≤ x ≤ **P**, and *a* can be any real number less than or equal to 0. The piece of trash is only considered thrown away when its center reaches the trash can's point, and it is not enough for some part of the piece of trash to just touch that point.

Bob is wondering: what is the largest piece of trash he can throw without hitting the ceiling or any obstacles? That is, we must find the maximum value of *R* for which there is at least one value *a* that satisfies the following: for any 0 ≤ x ≤ **P**, the Euclidean distance between (x, f(x)) and (x, H) is greater than *R*, and for each i, the Euclidean distance between the point (x, f(x)) and ($\mathbf{X_i}$, $\mathbf{Y_i}$) is greater than or equal to *R*.

Input

The input starts with one line containing one integer **T**, the number of test cases. **T** test cases follow. The first line of each test case contains three integers **N**, **P**, and **H**: the number of obstacles, the x-coordinate of the trash can, and the height of the ceiling. Then, there are **N** more lines; the i-th of those lines represents the i-th obstacle, and has two integers $\mathbf{X_i}$ and $\mathbf{Y_i}$, representing that obstacle's coordinates.

Output

For each test case, output one line `Case #x: y`, where x is the test case number (starting from 1) and y is a double representing the maximum radius **R**. Your answer will be considered correct if it is within an absolute or relative error of 10^{-4} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

1 ≤ **T** ≤ 50.
2 ≤ **P** ≤ 1000.
2 ≤ **H** ≤ 1000.
0 < $\mathbf{X_i}$ < **P**.
0 < $\mathbf{Y_i}$ < **H**.

Small dataset

N = 1.

Large dataset

1 ≤ **N** ≤ 10.

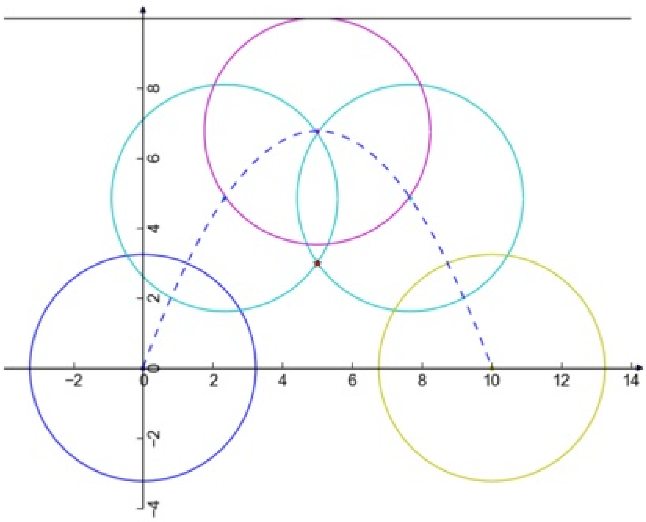
Sample

Input	Output

```
4             Case #1: 3.23874149472
1 10 10       Case #2: 4.00000000000
5 3           Case #3: 3.50000000000
1 10 10       Case #4: 2.23145912401
5 4
1 100 10
50 3
2 10 10
4 2
6 7
```

Note that the last sample case would not appear in the Small dataset.

The following picture illustrates Sample Case #1. Bob is at (0, 0), and the trash can is at (10, 0). There is a obstacle at point (5, 3), marked with a star. If Bob throws trash over the top of the obstacle, the maximal *R* is 3.2387, which requires an *a* of about -0.2705. If Bob throws trash under the obstacle, the maximal *R* is 3, which requires an *a* of 0. So the maximum *R* for this case is about 3.2387.



Sample Case #2 is like Sample Case #1, but the obstacle is one unit higher. Now, if Bob throws the trash under the obstacle, the maximal *R* is 4 (for *a* = 0). If he throws the trash over the obstacle, he can only use trash with a radius up to about 2.8306 (with *a* = -0.4). So the maximum *R* for this case is 4.