The 38^{th} ACM International Collegiate Programming Contest

Asia Chengdu Regional Contest

Problem List

Problem A	Assignment For Princess
Problem B	Beautiful Soup
Problem C	Clumsy Algorithm
Problem D	Dinner Coming Soon
Problem E	Exhausted Robot
Problem F	Fibonacci Tree
Problem G	GRE Words Revenge
Problem H	Hard Disk Drive
Problem I	ICPC Ranking
Problem J	Just Random

Problem A: Assignment For Princess

Description

Long long ago, in the Kingdom Far Far Away, there lived many little animals. And you are the beloved princess who is marrying the prince of a rich neighboring kingdom. The prince, who turns out to be a handsome guy, offered you a golden engagement ring that can run computer programs!

The wedding will be held next summer because your father, the king, wants you to finish your university first.

But you did't even have a clue on your graduation project. Your terrible project was to construct a map for your kingdom. Your mother, the queen, wanted to make sure that you could graduate in time. Or your wedding would have to be delayed to the next winter. So she told you how your ancestors built the kingdom which is called the Roads Principle:

1. Your kingdom consists of N castles and M directed roads.
2. There is at most one road between a pair of castles.
3. There won't be any roads that start at one castle and lead to the same one.

She hoped those may be helpful to your project. Then you asked your cousin Coach Pang (Yes, he is your troubling cousin, he always asks you to solve all kinds of problems even you are a princess.), the Minister of Traffic, about the castles and roads. Your cousin, sadly, doesn't have a map of the kingdom. Though he said the technology isn't well developed and it depends on your generation to contribute to the map, he told you the Travelers Guide, the way travelers describe the amazing road system:

1. No matter which castle you start with, you can arrive at any other castles.
2. Traveling on the M roads will take $1, 2, 3, \ldots, M$ days respectively, no two roads need the same number of days.
3. You can take a round trip starting at any castle, visiting a sequence of castles, perhaps visiting some castles or traveling on some roads more than once, and finish your journey where you started.
4. The total amount of days spent on any round trip will be a multiple of three.

But after a month, you still couldn't make any progress. So your brother, the future king, asked your university to assign you a simpler project. And here comes the new requirements. Construct a map that satisfies both the Roads Principle and the Travelers Guide when N and M is given.

There would probably be several solutions, but your project would be accepted as long as it meets the two requirements.

Now the task is much easier, furthermore your fiance sent two assistants to help you.

Perhaps they could finish it within 5 hours and you can think of your sweet wedding now.

Input

The first line contains only one integer T, which indicates the number of test cases.
For each test case, there is one line containing two integers N, M described above.
$(10 \leq N \leq 80, N + 3 \leq M \leq \frac{N^2}{7})$

Output

For each test case, first output a line "Case #x:", where x is the case number (starting from 1).

Then output M lines for each test case. Each line contains three integers A, B, C separated by single space, which denotes a road from castle A to castle B and the road takes C days traveling.

Oh, one more thing about your project, remember to tell your mighty assistants that if they are certain that no map meets the requirements, print one line containing one integer -1 instead.

Note that you should not print any trailing spaces.

Sample

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

Hint

The restrictions like $N \geq 10$ will be too big for a sample. So the sample is just a simple case for the detailed formats of input and output, and it may be helpful for a better understanding. Anyway it won't appear in actual test cases.

Problem B: Beautiful Soup

Description

Coach Pang has a lot of hobbies. One of them is playing with "tag soup" with the help of Beautiful Soup. Coach Pang is satisfied with Beautiful Soup in every respect, except the `prettify()` method, which attempts to turn a soup into a nicely formatted string. He decides to rewrite the method to prettify a HTML document according to his personal preference. But Coach Pang is always very busy, so he gives this task to you. Considering that you do not know anything about "tag soup" or Beautiful Soup, Coach Pang kindly left some information with you:

In Web development, "tag soup" refers to formatted markup written for a web page that is very much like HTML but does not consist of correct HTML syntax and document structure. In short, "tag soup" refers to messy HTML code.

Beautiful Soup is a library for parsing HTML documents (including "tag soup"). It parses "tag soup" into regular HTML documents, and creates parse trees for the parsed pages.

The parsed HTML documents obey the rules below.

HTML

- HTML stands for HyperText Markup Language.
- HTML is a markup language.
- A markup language is a set of markup tags.
- The tags describe document content.
- HTML documents consist of tags and texts.

Tags

- HTML is using *tags* for its syntax.
- A tag is composed with special characters: '**<**', '**>**' and '**/**'.
- Tags usually come in pairs, the *opening tag* and the *closing tag*.
- The *opening tag* starts with "**<**" and the *tagname*. It usually ends with a "**>**".
- The *closing tag* starts with "**</**" and the same tagname as the corresponding opening tag. It ends with a "**>**".
- There will not be any other angle brackets in the documents.
- Tagnames are strings containing only lowercase letters.
- Tags will contain no line break ('\n').
- Except tags, anything occured in the document is considered as *text content*.

Elements

- An element is everything from an opening tag to the matching closing tag (including the two tags).
- The element content is everything between the opening and the closing tag.
- Some elements may have no content. They're called *empty elements*, like `<hr></hr>`.
- Empty elements can be closed in the opening tag, ending with a "**/>**" instead of "**>**".
- All elements are closed either with a closing tag or in the opening tag.
- Elements can have attributes.
- Elements can be nested (can contain other elements).
- The `<html>` element is the container for all other elements, it will not have any attributes.

Attributes

- Attributes provide additional information about an element.
- Attributes are always specified in the opening tag after the tagname.
- Tag name and attributes are separated by single space.
- An element may have several attributes.
- Attributes come in `name="value"` pairs like `class="icpc"`.
- There will not be any space around the `'='`.
- All attribute names are in lowercase.

A Simple Example `<a href="http://icpc.baylor.edu/">ACM-ICPC</a>`

- The `<a>` element defines an HTML link with the `<a>` tag.
- The link address is specified in the href attribute.
- The content of the element is the text "ACM-ICPC"

You are feeling dizzy after reading all these, when Coach Pang shows up again. He starts to spout for hours about his personal preference and you catch his main points with difficulty. Coach Pang says:

Your task is to write a program that will turn parsed HTML documents into formatted parse trees. You should print each tag or text content on its own line preceded by a number of spaces that indicate its depth in the parse tree. The depth of the root of the a parse tree (the `<html>` tag) is 0. He is satisfied with the tags, so you **shouldn't** change anything of any tag. For text content, throw away unnecessary white spaces including space (ASCII code 32), tab (ASCII code 9) and newline (ASCII code 10), so that *word*s (sequence of characters without white spaces) are separated by single space. There should not be **any trailing space** after each line nor **any blank line** in the output. The line contains only white spaces is also considered as blank line. You quickly realize that **your only job is to deal with the white spaces**.

Input

The first line of the input is an integer T representing the number of test cases.

Each test case is a valid HTML document starts with a `<html>` tag and ends with a `</html>` tag. See sample below for clarification of the input format.

The size of the input file will not exceed 20KB.

Output

For each test case, first output a line "Case #x:", where x is the case number (starting from 1).

Then you should write to the output the formatted parse trees as described above. See sample below for clarification of the output format.

Sample

Input	*Output*
2 <html><body> ACM ICPC Hello World </body></html> <html><body> Asia Chengdu Regional ACM-ICPC </body></html>	Case #1: <html> ␣ <body> ␣␣ ␣␣␣ ACM ICPC ␣␣ ␣␣ ␣␣␣ Hello ␣␣␣ ␣␣␣ World ␣␣ ␣ </body> </html> Case #2: <html> ␣ <body> ␣␣ ␣␣␣ Asia Chengdu Regional ␣␣ ␣␣ ␣␣␣ ACM-ICPC ␣␣ ␣ </body> </html>

Hint

Please be careful of the number of leading spaces of each line in above sample output.

Problem C: Clumsy Algorithm

Description

Sorted Lists are thought beautiful while permutations are considered elegant. So what about sequence $(1, 2, \ldots, n)$? It is the oracle from god. Now Coach Pang finds a permutation $(p_1, p_2, \ldots, p_n)$ over $\{1, 2, \ldots, n\}$ which has been shuffled by some evil guys. To show Coach Pang's best regards to the oracle, Coach Pang decides to rearrange the sequence such that it is the same as $(1, 2, 3, \ldots, n)$. The time cost of swapping p_i and p_j is $2|i - j| - 1$. Of course, the minimum time cost will be paid since Coach Pang is lazy and busy. Denote the minimum time cost of the task as $f(p)$. But Coach Pang is not good at maths. He finally works out a clumsy algorithm to get $f(p)$ as following:

$$g(p) = \sum_{1 \le i \le n} \max(0, i - p_i)$$

Coach Pang's algorithm is clearly wrong. For example, $n = 3$ and $(3, 2, 1)$ is the permutation. In this case, $f(p) = 3$ but $g(p) = 0 + 0 + 2 = 2$. The question is that how many permutations p of $\{1, 2, \ldots, n\}$ such that $f(p) = g(p)$. To make the problem more challenge, we also restrict the prefix of p to $(a_1, a_2, \ldots, a_k)$. To sum up, you need to answer the question that how many permutations p of $\{1, 2, \ldots, n\}$ with the fixed prefix $p_1 = a_1, p_2 = a_2, \ldots, p_k = a_k$ such that $f(p) = g(p)$. Since the answer may be very large, for convenience, you are only asked to output the remainder divided by $(10^9 + 7)$.

Input

The first line contains a positive integer $T(1 \le T \le 100)$, which indicates the number of test cases. T lines follow. Each line contains $n, k, a_1, a_2, a_3, \ldots, a_k$. ($1 \le n \le 100, 0 \le k \le n, 1 \le a_i \le n$ and all a_i are distinct.)

Output

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

Sample

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

Hint

Among all permutations over $\{1, 2, 3\}$, $(3, 2, 1)$ is the only counter-example.

Problem D: Dinner Coming Soon

Description

Coach Pang loves his boyfriend Uncle Yang very much. Today is Uncle Yang's birthday, Coach Pang wants to have a romantic candlelit dinner at Uncle Yang's house and he has to arrive there in T minutes.

There are N houses in their city numbered from 1 to N. Coach Pang lives in house 1 while Uncle Yang lives in house N. The houses are connected by M directed roads. It takes some time and usually a fee to pass one road. Coach Pang wants to reach Uncle Yang's house before the dinner starts with as much money as possible.

But the matter is not so simple. Coach Pang decides to do some salt trade on the way to Uncle Yang's house. The host of each house offers a price of a bag of salt, so Coach Pang can make a profit from the price differences. Each time when Coach Pang arrives at a house (except the house 1 and the house N). He can buy **one** bag of salt, sell **one** bag of salt or do nothing. Coach Pang can carry at most B bags of salt with him, and he carries no salt when he leaves his house. The trading is so efficient that the time cost of trading can be ignored.

However, the problem is more complicated than imagine. Coach Pang has a handheld device that can perform a journey around K parallel universes numbered from 0 to $K-1$. Coach Pang lives in the universe 0. When Coach Pang uses the device in universe i, he will be transported to the same place and the same time of universe $(i+1)$ mod K. The host of the house at the same place in different universe may offer a different price of salt. Luckily, the time cost and fee of the city roads are uniform among the K universes. The journey between universes costs no time but Coach Pang has to stand still watching the ads on the device for one minute every time before the device works. Remember, Coach Pang should never visit house 1 or house N in a universe other than universe 0, because the situation might become uncontrollable if he bumps into himself or his boyfriend in another universe.

The time is running out. Coach Pang asks you to tell him whether he can arrive at Uncle Yang's house in time, and how much money Coach Pang can have at most when the dinner starts. Coach Pang has R *yuan* at the start, and will end his journey immediately once he arrives at Uncle Yang's house. He must arrive at Uncle Yang's house in T minutes, and he can't have negative amount of money anywhere anytime. Please help him!

Input

The first line of the input is an integer C representing the number of test cases.

For each test case, the first line will contain 6 integers N, M, B, K, R, T, as described above. $(2 \leq N \leq 100, 0 \leq M \leq 200, 1 \leq B \leq 4, 2 \leq K \leq 5, 0 \leq R \leq 10^5, 0 \leq T \leq 200)$

The following K lines contain N integers each, indicating the price p_{ij} $(0 \leq i < K, 1 \leq j \leq N)$ for a bag of salt offered by the host of house j in the universe i. The price of house 1 and house N will be marked as -1.$(1 \leq p_{ij} \leq 100)$

Then M lines follow, each contains 4 integers a, b, t and m, indicating that there is a road from house a to house b that costs t minutes of time and m *yuan* of money. $(1 \leq a, b \leq N, 1 \leq t \leq 15, 0 \leq m \leq 100)$

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is the most money Coach Pang can have if he can have dinner with Uncle Yang on time. Print "Forever Alone" otherwise.

Sample

Input	*Output*
2 3 2 1 2 10 6 -1 1 -1 -1 5 -1 1 2 1 0 2 3 1 1 2 2 1 2 5 5 -1 -1 -1 -1 1 2 10 2 1 2 2 10	Case #1: 17 Case #2: Forever Alone

Problem E: Exhausted Robot

Description

You want a tidy palace but you are too lazy to do the cleaning. As a result, your cousin Coach Pang gave you a cleaning robot. Unfortunately, the robot has some flaws and some furniture may hamper the cleaning.

To simplify the problem, we consider all objects in a 2D-Plane. The room is viewed as a rectangle with edges parallel to the axis. Both robot and furniture are in shape of convex polygons.

During the cleaning, the robot can move towards any direction (but only translation are permitted which means it cannot rotate). Although the robot has the ability to move through furniture, it can only do cleaning when it has no area outside the room or intersect with furniture. Besides, only one point can do the cleaning which is given as the first vertex of the robot in the input.

Input

The first line of the input contains an integer T indicates the number of test cases.

In the first line of each test case, there will be an integer n $(0 \leq n \leq 20)$ and then $(n+1)$ blocks of data describing the objects. The first n blocks for furniture and the last one for the robot. All the vertices of an object are shown in corresponding block with counter-clockwise order. The first line of each block contains an integer m $(3 \leq m \leq 20)$. Then m lines follow. In each line, there are two integers x_i, y_i indicating a vertex of the convex polygon.

At the end of each set of data, there will be four integers in a line, $x_{BL}, y_{BL}, x_{TR}, y_{TR}$, indicates the coordinates of the bottom left corner and the top right corner of your room.

The absolute value of all coordinates are within 10^3.

Output

For each test case, output one line "Case #x: y", where x is the case number (starting from 1) and y is size of the area that robot can clean, rounded to 3 digits after decimal point.

Sample

Input	*Output*
2 1 4 3 3 4 3 4 4 3 4 4 1 1 2 1 2 2 1 2 0 0 10 10 0 4 1 1 2 1 2 2 1 2 0 0 10 10	Case #1: 77.000 Case #2: 81.000

Problem F: Fibonacci Tree

Description

Coach Pang is interested in Fibonacci numbers while Uncle Yang wants him to do some research on Spanning Tree. So Coach Pang decides to solve the following problem:

Consider a bidirectional graph G with N vertices and M edges. All edges are painted into either white or black. Can we find a Spanning Tree with some positive Fibonacci number of white edges? (Fibonacci number is defined as $1, 2, 3, 5, 8, \cdots$)

Input

The first line of the input contains an integer T, the number of test cases.

For each test case, the first line contains two integers $N(1 \leq N \leq 10^5)$ and $M(0 \leq M \leq 10^5)$.

Then M lines follow, each contains three integers u, v $(1 \leq u, v \leq N)$ and c $(0 \leq c \leq 1)$, indicating an edge between u and v with a color c (1 for white and 0 for black).

Output

For each test case, output a line "Case #x: s". x is the case number and s is either "Yes" or "No" (without quotes) representing the answer to the problem.

Sample

Input	*Output*
2 4 4 1 2 1 2 3 1 3 4 1 1 4 0 5 6 1 2 1 1 3 1 1 4 1 1 5 1 3 5 1 4 2 1	Case #1: Yes Case #2: No

Problem G: GRE Words Revenge

Description

Now Coach Pang is preparing for the Graduate Record Examinations as George did in 2011. At each day, Coach Pang can:

- "+w": learn a word w
- "?p": read a paragraph p, and count the number of learnt words. Formally speaking, count the number of substrings of p which is a learnt words.

Given the records of N days, help Coach Pang to find the count. For convenience, the characters occured in the words and paragraphs are only '0' and '1'.

Input

The first line of the input file contains an integer T, which denotes the number of test cases. T test cases follow.

The first line of each test case contains an integer N ($1 \leq N \leq 10^5$), which is the number of days. Each of the following N lines contains either "+w" or "?p". Both p and w are 01-string in this problem.

Note that the input file has been *encrypted*. For each string occured, let L be the result of last "?" operation. The string given to you has been shifted L times (the shifted version of string $s_1 s_2 \ldots s_k$ is $s_k s_1 s_2 \ldots s_{k-1}$). You should *decrypt* the string to the original one before you process it. Note that L equals to 0 at the beginning of each test case.

The test data guarantees that total length of the words does not exceed 10^5 and total length of the paragraphs does not exceed $5 \cdot 10^6$.

Output

For each test case, first output a line "Case #x:", where x is the case number (starting from 1). And for each "?" operation, output a line containing the result.

Sample

Input	*Output*
2 3 +01 +01 ?01001 3 +01 ?010 ?011	Case #1: 2 Case #2: 1 0

Problem H: Hard Disk Drive

Description

Yesterday your dear cousin Coach Pang gave you a new 100MB hard disk drive (HDD) as a gift because you will get married next year.

But you turned on your computer and the operating system (OS) told you the HDD is about 95MB. The 5MB of space is missing. It is known that the HDD manufacturers have a different capacity measurement. The manufacturers think 1 "kilo" is 1000 but the OS thinks that is 1024. There are several descriptions of the size of an HDD. They are byte, kilobyte, megabyte, gigabyte, terabyte, petabyte, exabyte, zetabyte and yottabyte. Each one equals a "kilo" of the previous one. For example 1 gigabyte is 1 "kilo" megabytes.

Now you know the size of a hard disk represented by manufacturers and you want to calculate the percentage of the "missing part".

Input

The first line contains an integer T, which indicates the number of test cases.

For each test case, there is one line contains a string in format "number[unit]" where number is a positive integer within $[1, 1000]$ and unit is the description of size which could be "B", "KB", "MB", "GB", "TB", "PB", "EB", "ZB", "YB" in short respectively.

Output

For each test case, output one line "Case #x: y", where x is the case number (starting from 1) and y is the percentage of the "missing part". The answer should be rounded to two digits after the decimal point.

Sample

Input	*Output*
2 100[MB] 1[B]	Case #1: 4.63% Case #2: 0.00%

Hint

In the first example, the manufacturers think 100MB = 10^5KB = 10^8B, but the OS thinks 100MB = 100×2^{10}KB = 100×2^{20}B. So $1 - \frac{10^8}{100 \times 2^{20}} = 4.63\%$ is missing.

Problem I: ICPC Ranking

Description

There are so many submissions during this contest. Coach Pang can not determine which team is the winner. Could you help him to print the score board?

As we all know, the contest executes by the teams submit codes for some problems. Simply, we assume there are three kinds of results for every submission.

ERROR There was something wrong. The team didn't solve the problem but won't get any penalty.

NO Sorry, the code was not right. The team didn't solve the problem.

YES Yeah, AC. The team solved the problem.

To make the contest more exciting, we set a time called frozen time. If one team doesn't solve one of the problems before the frozen time and submits at least one submission on this problem after or exactly at the frozen time, this problem of this team is called frozen. For different team, the frozen problems will be different. For the frozen problems, the score board will only show how many submissions the team has been submitted but won't show the result of the submissions.

The rank is determined by the following factors. Remember, we only consider the unfrozen problems. The frozen problems will be ignored. The following factors are ordered with the priority from high to low.

Solved The team who solves more problems will place higher.

Penalty The team who gets less penalty time will place higher. Only solved problems will give penalty time. Every solved problem will give $T + 20X$ penalty time. T is the time of the first YES, X is the number of NOs before the first YES.

Last Solved The team who solved their last problem earlier will place higher. If there is a tie, we compare the second last problem, then the third last problem, etc.

Name The team whose name comes later in lexicographical order will place higher.

At the end of the contest, the score board will be unfrozen. First, choose the team which has frozen problems with the lowest rank. Then choose one frozen problem of this team. If the team has multiple frozen problems, choose the first frozen problem in alphabetic order. Then show the result of the problem, recalculate the rank, change the score board and make the problem unfrozen for this team. Repeat this procedure until no teams have frozen problems. Then we get the final score board.

Please help Coach Pang to print the initial score board, the final score board and the process of the unfreeze procedure.

Input

The first line contains an integer C, which indicates the number of test cases.

For each test case, the first line will have four integers n, m, T and t. $n(1 \leq n \leq 50000)$ is the number of submissions. $m(1 \leq m \leq 26)$ is the number of problems. $T(1 \leq T \leq 10000)$ is the total time of the contest. $t(0 \leq t \leq T)$ is the frozen time.

The following n lines, each line will in the form "`Name Problem Time Result`". `Name` is the team's name which only contains letters and digits with at most 20 characters. `Problem` is the identifier of the problem which is a capital letter from A to the m-th letter. `Time` is a integer indicates the submission time which greater or equal then 0 and less then T. `Result` is one string which equal YES, NO or ERROR.

Every team will have at least one submission. If one team has multiple submissions at the same time, we consider the ERRORs come before NOs and NOs come before YESs.

Output

For each test case, first print "Case #x:", x is the case number start from 1.

Then print the initial score board (before the unfreeze procedure) ordered by the rank. For every team, print "`Name Rank Solved Penalty A B C ...`". `Name` is the team's name. `Rank` is the team's rank. `Solved` is the number of solved problems. `Penalty` is the team's penalty time. `A B C ...` is the condition of every problem is the format below:

+x The problem is unfrozen and solved. x is the number of NOs before the first YES. If $x = 0$, print "`+`" instead of "`+0`".

-x The problem is unfrozen but not solved. x is the number of NOs. If $x = 0$, print "`.`" instead of "`-0`".

-x/y The problem is frozen. x is the number of NOs before the frozen time. y is the number of submissions after or exact at the frozen time. If $x = 0$, print "`0/y`" instead of "`-0/y`".

After the initial score board, print the process of the unfreeze procedure. During the unfreeze procedure, every time one team unfreezes one frozen problem and causes the change of its rank, print "`Name1 Name2 Solved Penalty`". Suppose this team is team A. `Name1` is the name of team A. `Name2` is the name of the team which is overtaken by team A with the highest rank. `Solved` is the new number of the solved problems of team A. `Penalty` is the new penalty time of team A.

Finally print the final score board (after the unfreeze procedure) as the same format as the initial score board.

See the sample to get more details.

Sample

Input

```
1
20 12 300 240
Epic B 12 YES
Epic A 14 NO
Rivercrab E 25 YES
Two2erII B 100 NO
Epic A 120 YES
Rivercrab I 150 NO
Two2erII C 160 NO
Epic C 180 YES
Two2erII C 180 NO
Rivercrab F 226 YES
Two2erII C 230 YES
Two2erII L 241 YES
Epic F 246 YES
Epic G 260 YES
Rivercrab I 289 YES
Epic D 297 YES
Musou H 299 YES
Musou I 299 YES
Musou J 299 YES
Musou K 299 YES
```

Output

```
Case #1:
Epic 1 3 332 +1 + + 0/1 . 0/1 0/1 . . . . .
Rivercrab 2 2 251 . . . . + + . . -1/1 . . .
Two2erII 3 1 270 . -1 +2 . . . . . . . . 0/1
Musou 4 0 0 . . . . . . . 0/1 0/1 0/1 0/1 .
Musou Two2erII 2 598
Two2erII Musou 2 511
Musou Rivercrab 3 897
Rivercrab Musou 3 560
Musou Epic 4 1196
Epic Musou 4 629
Epic 1 6 1135 +1 + + + . + + . . . . .
Musou 2 4 1196 . . . . . . . + + + + .
Rivercrab 3 3 560 . . . . + + . . +1 . . .
Two2erII 4 2 511 . -1 +2 . . . . . . . . +
```

Problem J: Just Random

Description

Coach Pang and Uncle Yang both love numbers. Every morning they play a game with number together. In each game the following will be done:

1. Coach Pang randomly choose a integer x in $[a, b]$ with equal probability.
2. Uncle Yang randomly choose a integer y in $[c, d]$ with equal probability.
3. If $(x + y)$ mod $p = m$, they will go out and have a nice day together.
4. Otherwise, they will do homework that day.

For given a, b, c, d, p and m, Coach Pang wants to know the probability that they will go out.

Input

The first line of the input contains an integer T denoting the number of test cases.

For each test case, there is one line containing six integers a, b, c, d, p and m ($0 \leq a \leq b \leq 10^9, 0 \leq c \leq d \leq 10^9, 0 \leq m < p \leq 10^9$).

Output

For each test case output a single line "Case #x: y". x is the case number and y is a fraction with numerator and denominator separated by a slash ('/') as the probability that they will go out. The fraction should be presented in the simplest form (with the smallest denominator), but always with a denominator (even if it is the unit).

Sample

Input	*Output*
4 0 5 0 5 3 0 0 999999 0 999999 1000000 0 0 3 0 3 8 7 3 3 4 4 7 0	Case #1: 1/3 Case #2: 1/1000000 Case #3: 0/1 Case #4: 1/1