AB BANK IUT 5th NATIONAL

ICT FEST

4th & 5th October, 2013

Organized by

IUTCS

IUT Computer Society

IUT Inter University Programming Contest 2013

Problem setters' List

Abu Obaida Opu, Eyeball Networks
Anindya Das, Iowa State University
Anna Fariha, UIU
F.A. Rezaur Rahman Chowdhury, Eyeball Networks
Md. Moshiur Rahman, Eyeball Networks
Md. Shiplu Hawlader, UAP
Md. Towhidul Islam Talukder, SDSL
Mir Wasi Ahmed, MuktoSoft
Muhammed Hedayet, Zahan Soft
Tasnim Imran Sunny, SDSL

A An Interesting Game

Jodu and Kodu are two best friends and they like to play different games with each other. One day Jodu discovered an interesting two player game and asked Kodu to participate in the game.

Jodu has **N** marbles each of unique weights from **1** to **N** and placed them in ascending order in front of them. At first move, Jodu will select **K** consecutive marbles **(X, X+1, X+2 … X+K-1)** starting from **X** and will remove them. Kodu will do the same as the second move of the game. Before the move Kodu wants to calculate, what is the total weight of the remaining marbles after Jodu's move. But Kodu is badly weak in mathematics, he don't know how to calculate sum of consecutive numbers.

Write a program to help Jodu and Kodu to find the total weight remains after the first move of Jodu.

Input

First line of the input will contain an integer **T(T<100)**, number of test cases to follow. Each of the next **T** lines will contain three integers **N, K, X** **($0<N<10^4$, 1≤K≤N, 1≤X, X+K-1≤N)** .

Output

For each input, print the output in the format, **Case X: Y** (here, **X** is the test case number starting from 1 and **Y** is the answer).

Sample Input	Output for Sample Input

2
5 2 2
6 3 1
	Case 1: 10
	Case 2: 15

B Barbarian Horde

Once upon a time, there was a clan named Barbarian Horde, they called it BH in short. There were **8** members in the clan: BH.BloodThirsY, BH.CrusniK, BH.ZorredekA, BH.AnsweR, BH.NobodY, BH.ShadowKnighT, BH.NomaD, BH.TyraNt.S. They used to visit the village in disguise to protect them from evils. Each night they took disguise of various heroes. Some heroes were agile, some were strong and some were wise! But they sometimes became confused to decide which hero they should choose. There were many heroes to choose from, but they had a rule, each of the chosen heroes must have names starting with a unique letter. So it was very tough for them to choose a hero by themselves, as the chosen hero may conflict with some other chosen hero by another member of the clan. So, one day, the lord of the clan chose some letters to represent the hero names. For example, if the king gave "dwovmqsh", 'd' stands for Drow Ranger, 'w' for Wind Runner, 'o' for Omni Knight, 'v' for Vengeful Spirit, 'm' for Mirana ,'q' for Queen of Pain, 's' for Shadow Fiend, 'h' for Huskar. The hero having the **first letter of its name at** earlier position in English alphabet are considered **more powerful** than the hero having first letter of its name at **later position** in English alphabet. All of the clan members were greedy to take the most powerful hero.

To resolve this, the Queen of the clan proposed an idea. She made a chart representing all the anagrams of the word given by the lord. Then she kept the anagrams that are **wavy anagrams**. **Wavy anagram**s are the anagrams of the original word, in which, the 1st hero is more powerful than the 2nd hero, the 2nd hero is less powerful than the third hero, the third hero is more powerful than the 4th hero, the 4th hero is less powerful than the 5th hero and so on.

This was done so that everyone is happy. The chosen anagrams are kept sorted lexicographically in a special box, which had many cells; each cell contained one **wavy anagram** of the word chosen by the BH Clan Lord. On the **N-th** day, the **N-th** cell is chosen and the **N-th** lexicographically smallest **wavy anagram** is retrieved from that cell.

You are given the Lord's chosen word **W** and the day number **N**; your job is to find the **wavy anagram** for that day to assign heroes to BH clan members.

Input

The first line of the input denotes **T (≤1000)**, the number of test cases to follow. Each test case consists of a string **W (0 < |W| < 24)** containing only lower case letters of English alphabet with no repeated characters and a number **N** (fits in **64 bit signed integer**). The input is such that there is always a solution.

Output

For each input, you have to print the case number first, followed by **N-th** lexicographically smallest **wavy anagram** of the given string **W**. See the sample input output for more details.

Sample Input	Output for Sample Input
4 take 1 take 2 take 3 take 4	Case 1: aket Case 2: atek Case 3: ekat Case 4: etak

C Countree Song

Professor Heickal absolutely hates cycles. So when it comes to graphs, he is partial to trees and forests. Not surprisingly, his favorite band is 'Porcupine Tree', favorite movie is the 'Forrest Gump' and he loves eating Black Forests. Anyway, in one evening, I went to him to discuss about problems concerning people, country, the world and the humankind.

In the middle of our discussion, he suddenly said, "See! Everybody loves cycles! People like to live the life their ancestors lived, the world leaders never learn from their mistakes, people cut down trees and destroy forests all the time and Professor Hawlader keeps on giving people amulets to change their fortunes! These kinds of things are recurring all the time! You got to stop these recurrences by putting a base case. Scientists say, Dynamic Programming is very important for life. Everyone knows that trees, forests and directed acyclic graphs are suitable for DP... ". At that point, he noticed that I was nervously checking my watch. He changed the topic and said, "Let's solve an interesting problem! And it's about... trees!"

The problem was like this: Given the number of nodes at different depths of a rooted tree, count the number of valid trees possible with that configuration. Formally, you will be given an array **C** with C_i = number of nodes at depth **i** (explanations of the terms is given afterwards). Each node of the tree is labeled with a unique integer. The nodes are labeled in this way:

Let, $\mathbf{S_i = C_0 + C_1 + \ldots + C_i}$
The root is labeled with **1**.
The nodes at depth **1** are labeled by integers from **2** to $\mathbf{S_1}$ (inclusive)
...
...
The nodes at depth **d** are labeled by integers from $\mathbf{S_{d-1}+1}$ to $\mathbf{S_d}$ (inclusive)

You'll need to find out the number of **k-ary** trees with **C** array.

A **tree** is a connected graph without a cycle. A tree is called a **rooted tree** if one vertex has been designated the root, in which case the edges have a natural orientation, away from the root. The **depth** of a node is the length of the path to its root. A **k-ary tree** is a rooted tree in which each node has no more than **k** children.

Anyway, as I was severely disoriented by that discussion with the professor, I could not solve the problem that time. What about you?

Input

The first line of the input will contain the number **T**, the number of test cases. It will be followed by **T** sets of inputs. Each set of test case will have two lines. First line will contain pair of integers **d**, depth of the tree and **k**, as defined in the statement**.** In the next line, there will be **d+1** integers, giving the **C** array where **i-th** integer will be the value of $\mathbf{C_i}$.

Constraints

$1 \le T \le 256$

$0 \le d \le 512$

$0 \le k \le 8$

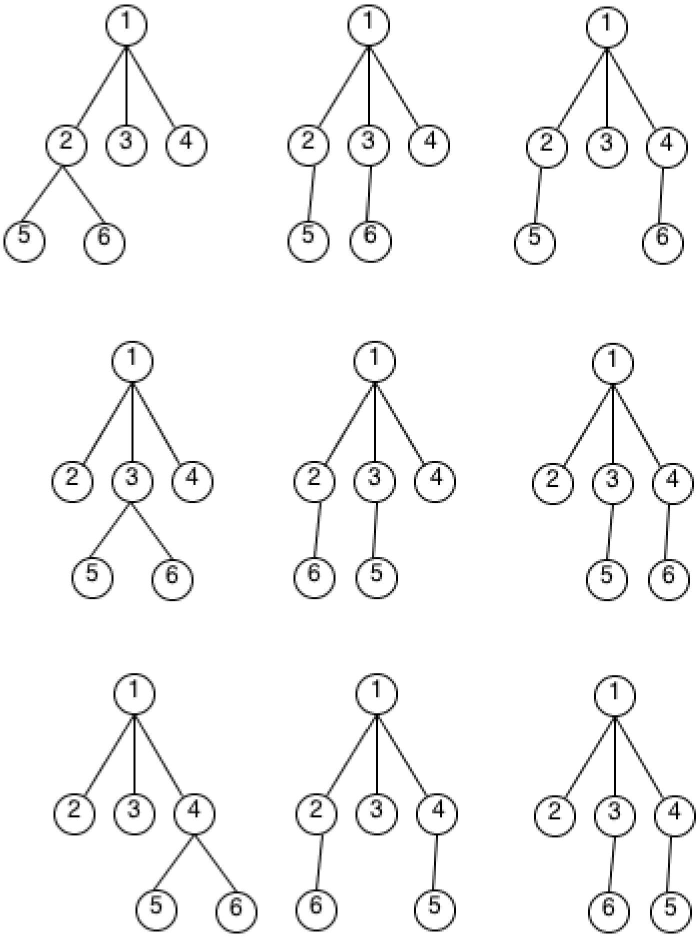
$0 \le C_i \le 512$ ($1 \le i \le d$) and $C_0 = 1$

Output

For each set of input, print the output in the format **Case X: Y** (here, **X** is the serial of the input and **Y** is the answer) in a line. As the output can be very large, print the answer modulo **1000000009**.

Sample Input	Output for Sample Input

2
2 3
1 3 2
8 8
1 4 10 14 17 24 26 26 30
	Case 1: 9
	Case 2: 23879694



D Display Problem

Nut and Boltu want to buy a LCD display. They went to the market to buy the display. The salesman showed them different types of displays. But he did not tell them the size of the displays. He only showed them the letter **8** printed in the display. Being very inexperienced about LCD displays Nut and Boltu wants your help to find the dimension of the display board. There are some bad displays too. You need to find them as well.

A display can be represented as a rectangular grid of some special characters. A **good display** is a display in which **8** is printed in a way so that the display has a **degree**. The **degree** of a display is a positive integer **D** such that:

1. The number of rows of the grid is **2D+3**.
2. The number of columns of the grid is **D+2**.
3. In **0-th row** (0-based): the **0-th** column and **(D+1)-th** column contains dot(**.**) and other columns contain hyphens(**-**).
4. In **(D+1)-th row** (0-based): the **0-th** column and **(D+1)-th** column contains dot(**.**) and other columns contain hyphens(**-**).
5. In (**2D+2**)-th **row**(0-based): the **0-th** column and **(D+1)-th** column contains dot(**.**) and other columns contain hyphens(**-**).
6. In every other row: the **0-th** column and **(D+1)-th** column contains bar(**|**) and other columns contain dot(**.**).

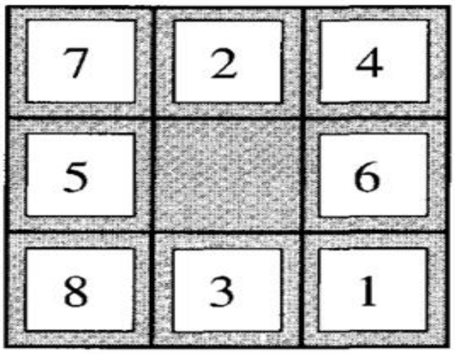
Input
First line of the input contains the number of test cases, **T** (**T ≤ 50**). In each test case **i-th** line contains **i-th** row of the grid that defines the display, A line containing only an asterisk(*****) marks the end of test case. Grid will contain only characters dot(**.**), bar(**|**) and hyphen(**-**). There will be at most **25** lines in each test case and a single line will be **non-empty** and contain **at most 30** characters. There will be at least one line in every test case.

Output
For each input, print the output in the format, **Case X: Y**(here **X** is the serial of the input and **Y** is the degree of the grid if the display is good otherwise print **'Bad'**).

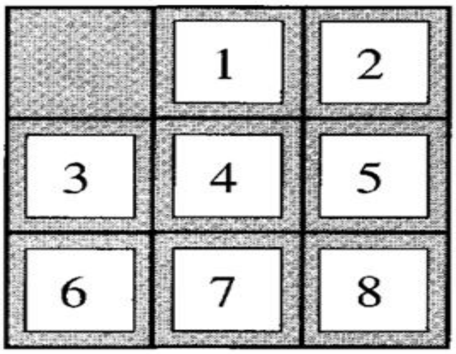
<table>
<tr><th>Sample Input</th><th>Output for Sample Input</th></tr>
<tr><td>2
.---.
|...|
|...|
|...|
.---.
|...|
|...|
|...|
.---.
*
.---.
|...|
|...|
|...|
.---.
|...|
|...|
.---.
*</td><td>Case 1: 3
Case 2: Bad</td></tr>
</table>

E Easy Puzzle

We know that the 8-puzzle is a sliding puzzle where there are numbered tiles in no particular order and only one square is empty. In each move, we can slide one **adjacent** tile to the empty space. Thus we need to place all the tiles in order. An example of initial state and goal state is shown here.



Start State



Goal State

Here, we are considering an easier version of the puzzle. In each move, any tile can be moved to the empty space, i.e. the **adjacency is not required**. In other words, we are allowed to swap the position of any tile with the empty space in one move. The goal is to calculate the minimum number of moves to solve the puzzle for a **N x N** board.

Input

The first line of input contains a positive integer **T (T<100)**. Then **T** cases follow, where the first line of each case contains a positive integer **N (2 ≤ N ≤ 500)**. Then each of the following **N** lines contains **N** space separated integers denoting a row of the puzzle. The empty square is denoted by **0**. Each integer from **0** to **N^2-1** (inclusive) will be present exactly once.

Output

For each case, print **Case X: Y** in a separate line**,** where **X** is the case number and **Y** is the minimum number of moves to solve the puzzle.

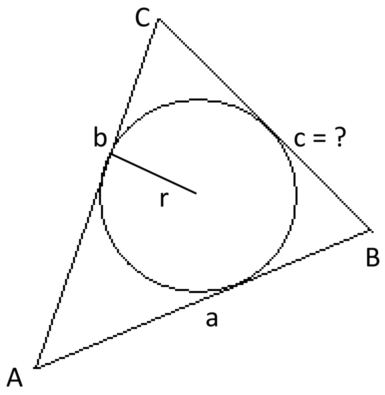
Sample Input	Output for Sample Input

2
3
7 2 4
5 0 6
8 3 1
2
3 0
1 2
	Case 1: 8
	Case 2: 3

F Fallen Inferno!

Yes, it might sound crazy to some but Mr. Fallen has fallen into *inferno. Inferno is not a good place to live so he doesn't have much time to waste. There is only one way to bring him out of inferno and the way is to answer a simple question asked by the devils. He has been given the length of two sides of a triangle ABC and the radius of the **incircle of the given triangle. Now he has to find the length of the other side of the given triangle.

But as you all might already have known, Fallen is good for nothing and that is why he falls here and there (But this time it's serious, Inferno is no good place to fall at all!) so he sends a message to the contest organizers to engage you, brilliant minds to help him.



Input

The first line of the input contains an integer **T(≤10,000**) denoting the number of test cases. Each of the following **T** lines consists of three positive integers **a**, **b**, **r(1≤ a, b, r ≤**10^9). Where **a=AB** and **b=AC** are the two sides of the triangle and **r** is the radius of the inner circle.

Output

For each input, print the output in the format, **Case X: Y**(here, **X** is the serial of the input and **Y** is the length of side BC). If there is multiple such BC, print the minimum one. If the triple **a**, **b** and **r** does not produce any valid **BC** print **-1**. Error less than **1e**$^{-6}$ or **0.000001** will be ignored.

Sample Input	Output for Sample Input

3
6 5 9
3 4 1
8 20 1
	Case 1: -1
	Case 2: 3.82842712
	Case 3: 12.42505840

*Inferno: Italian for "The Hell"

In geometry, the **incircle or **inscribed circle** of a triangle is the largest circle contained in the triangle; it touches (is tangent to) the three sides [Source: Wikipedia]

G Gotham’s Rail Track

Gothams railway track is made using rail blocks connected using joint bars. A rail block is made using two parallel rail and perpendicularly laid sleepers. In rail tracks a rail block can be connected to at most two other rail blocks. Several rail blocks are connected using joint bars to create a rail track.

In this problem you are going to work with **N** rail blocks numbered from **1 to N** and will be given following three types of queries:

1 u v – **connect** block **u** and **v** **(1 ≤ u,v ≤N and u ≠ v)** (any moment a block will be connected to at most two blocks).

2 u v – **disconnect** block **u** and **v** (it is ensured that this query will only disconnect existing connections). Two blocks **u** and v is considered connected if and only if there was a “**1 u v**” or “**1 v u**” query performed and no “**2 u v**” or “**2 v u**” query is performed after that.

3 u v – output the **longest distance** between **u** and **v**, distance between two blocks is equal to number of rail blocks in a path from **u** to **v** (**including u, v**). **If there is no path then output -1**.

Input

Input starts with an integer **T (T≤5)** denoting the number of test cases. First line of each test case contains two integers **N (2≤N≤10^5)** and **Q (1≤Q≤10^5)**. The next **Q** lines contain queries as described above.

Output

For each case print the case number in the first line. Then for each query **3 u v** print the answer in separate line. See sample input output for more details.

Sample Input	Output for Sample Input

1
5 6
1 1 2
1 3 4
1 1 3
3 1 4
2 3 4
3 1 4
	Case 1:
	3
	-1

H Help Gollum

Finally, desperate Bilbo asked Gollum the following programming puzzle. Help the poor creature with his dinner!

A = {A[1], A[2], … A[N]} is a sequence of lowercase letters.

B = {B[1], B[2], … B[K]} is another such sequence. B is called a **subsequence** of A if there exists a set of integers **S = {S[1], S[2], … S[K]}** such that the following two conditions are true:

(i) $1 \le S[1] < S[2] \ldots < S[K] \le N$

(ii) $A[S[i]] = B[i]$ for all $1 \le i \le K$

Here, **S** is called an **occurrence** of **B** in **A**. **S** is called the earliest occurrence of **B** in **A**, if there is no other occurrence **Y** such that, $Y[j] < S[j]$ for some $1 \le j \le K$.

The earliest occurrence **S** of **B** in **A** is called a **weak occurrence** if, $S[i+1] - S[i] \le M$ for all $1 \le i < K$. Here, **M** is called the weakness limit.

For example, if **M** = 2 then {b, c, d} has a weak occurrence in {a, **b,** y, **c, d,** c, d}, but {a, c, d} doesn't.

You are given a forbidden sequence **F** of lowercase letters and a weakness limit **M**. A sequence **X** of length **N** is called **strong** if one of the following conditions is true:

(i) the earliest occurrence of **F** in X is not a **weak occurrence** or

(ii) **F** doesn't occur in **X** at all.

Write a program to calculate the number of **strong** sequences of lowercase letters of length N. Print the answer modulo **1000000007**.

Input

The first line of the input contains **T**, the number of test cases. **T ≤ 5000**. Each test case consists of two lines. The first line contains a non-empty string of lowercase letters that denotes the forbidden sequence **F** which contains no more than **100** characters. The next line contains two positive integers **M** ($1 \le M \le 10$)and **N** ($1 \le N \le 10^9$), where **M** denotes the weakness limit and **N** denotes the desired length of the strong sequences.

Output

For each set of input, print the output in the format, **Case X: Y** where X is the serial of the input and **Y** is the desired output (see the sample output for clarification).

Sample Input	Output for Sample Input
2 ab 2 4 ww 1 2	Case 1: 453750 Case 2: 675

I Increasingly Strict Sequence

Given a sequence of **N** integers **A** = **{A[1], A[2] .. A[N] }** and all of the integers has an equal number of digits. You are allowed to perform the following operation on the sequence as many times as you wish:

Change a single digit from some integer into a different digit.You goal is to obtain a **strictly increasing** sequence where none of the numbers has a leading zero.

What is the minimum number of operations required to achieve this goal? Note that, the input sequence can have numbers with leading zeros.

Input

The first line contains an integer **T** (**T ≤ 30)** denoting the number of test cases.. The first line of each test case contains **N**(**1 ≤ N ≤50)** denoting the number of integers in the input sequence. Each of the next **N** lines contains an integer containing at most **50** digits. The numbers will contain **equal number** of digits.

Output

For each test case, print the answer in the format **Case X: Y**, where X is the serial of the input and Y is the minimum number of operations required to convert the input sequence into a strictly increasing sequence. If it is not possible to achieve this goal, print **-1**

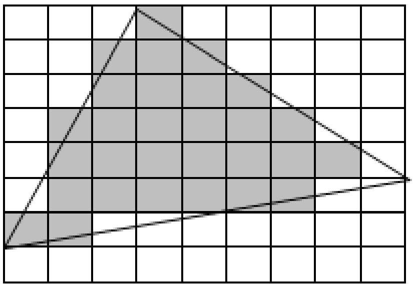
Sample Input	**Output for sample input**

`2`
`3`
`31`
`21`
`11`
`2`
`135`
`100`
	`Case 1: 2`
	`Case 2: 1`

J Just A Triangle

You are given the co ordinates of a triangle lying on a **2D** Cartesian plane. The whole plane can be divided into square blocks of **1 x 1** size. Your job is to find out how many of the **1 x 1** square blocks have **at least 50%** of its area **inside the triangle**.

The picture below shows a triangle where the vertices are at **(0, 1), (9, 3)** and **(3, 8).** And the shaded squares are the squares with **at least 50%** of its area inside the triangle.



Input

The first line of the input contains an integer **T(≤100**) denoting the number of test cases. Each of the following **T** lines contain six space separated integers $\mathbf{x_1\ y_1\ x_2\ y_2\ x_3\ y_3}$ giving the co ordinates of the triangle. The given co ordinates will form a valid triangle with positive area and all the co ordinates will be integers having values between **0** and **100** (inclusive).

Output

For each input, print the output in the format, **Case X: Y**(here, **X** is the serial of the input and **Y** is the number of squares which have **at least 50%** of its area inside the triangle).

Sample Input	Output for Sample Input

2
0 1 9 3 3 8
1 1 4 1 1 4
	Case 1: 29
	Case 2: 6