A — Area

Time Limit: 1 sec
Memory Limit: 32 MB

Steve has a dream to move from dormitory where we live now to a private house. He has already earned some money by solving various problems and now wants to make a first move towards his dream. Steve is going to buy a plot for his future house. However, due to high tax rate, he is going to buy only one plot. Certainly Steve wants to maximize its area for the amount of money he owns (and if there are several plots of same area, Steve certainly chooses a cheaper one). Can you help Steve?

The whole district where Steve wants to reside is divided into $N * M$ equal strips. Each strip is sold separately, for its own price $P_{i;j}$. According to the country law, plot is defined as a rectangular collection of strips with sides parallel to the district.

INPUT

There is a number of tests T ($T \leq 110$) on the first line. After T tests follows. Each test case is defined by three numbers NMK ($N; M \leq 100; K \leq 10^9$). Next N lines with M numbers each stands for prices of the strips $P_{i;j}$ ($P_{i;j} \leq 10^6$).

OUTPUT

For each test print a line formatted as `"Case #T: S P"`, where T stands for test case number (starting from 1), S for maximal plot area that Steve can afford and P for plot's total cost.

SAMPLE INPUT

```
1
5 6 15
10 1 10 20 10 10
30 1 1 5 1 1
50 1 1 20 1 1
10 5 5 10 5 1
40 10 90 1 10 10
```

SAMPLE OUTPUT

```
Case #1: 6 10
```

B — Arithmetic

Time Limit: 1 sec
Memory Limit: 32 MB

Alice is really bad at arithmetic. Sometimes she can't properly add two single-digit numbers without using a computer! It is quite embarrassing, so she comes up with silly explanations when someone spots an error. Last time, after writing $5 + 4 = 10$, Alice said - ”Hey, this is right in base 9!”.

Since you know how to program, can you help Alice and check her last bunch of calculations for any mistakes she might have left? She considers expression ”correct” if equality is satisfied for any positional numeral system with base B ($1 \leq B$) (in base 1 character `'1'` is used as only possible digit).

INPUT

The number of expressions N ($T \leq 100$) is given on the first line. Following `N` lines each have an expression in format $A + B = C$, where A, B, C ($0 \leq A + B, C \leq 10^5$) are non-negative decimal integers.

OUTPUT

For each expression, output a single line with the smallest counting system base B in which expression is correct. In case there is no such base, output `"0"`.

SAMPLE INPUT

```
5
155 + 102 = 301
1022 + 221 = 1303
6502 + 6800 = 11202
515 + 7 = 522
```

SAMPLE OUTPUT

```
6
4
0
10
10
```

C — Battleships

Time Limit: 1 sec
Memory Limit: 32 MB

Battleships game is a pen and paper game that was invented by Clifford Von Wickler in the early 1900s. In this game each player uses two N x N grids. One to arrange his ships and record the shots of the opponent. On the other grid the player records his own shots. Ships in battleship game can vary in size from 1 x 1 to 1 x $\frac{N}{2}$ and can be placed both vertically and horizontally. When all of the ship's cells have been hit, the ship is considered sunk, otherwise it is still "alive". Beside this, there can be more than one ship of each size, however none of two ships can overlap or touch.

In this problem you will be given the placement of ships on the player's grid. You will have to calculate the number of ships that the player still owns.

INPUT

There is a number of tests T ($T \leq 100$) on the first line. Each test case contains a positive number N ($N \leq 100$) — grid size. Next N lines contain N characters each, describing the playing grid. Character "`.`" stands for an empty cell, "`x`" for a cell containing a ship or its part and "`@`" for already hit part of a ship.

OUTPUT

For each test case output a single line "`Case T: N`". Where T is the test case number (starting from 1) and N is the number of still "alive" ships.

SAMPLE INPUT

```
2
4
x...
..x.
@.@.
....
2
..
x.
```

SAMPLE OUTPUT

```
Case 1: 2
Case 2: 1
```

D — Binary Calculator

Time Limit: 1 sec
Memory Limit: 32 MB

This problem doesn't have a story. You just have to write a calculator which could perform simple binary operations. You will be given an expression and your program should calculate it's result. Expression is formed according to this rules:

```
<digit> ::= 0|1
<number> ::= <digit>|<digit><number>
<unary_operator> :== not | shr | shl
<binary_operator> :== xor | and | or
<token> ::= <number> | <unary_operator> <token> | <token> <binary_operator> <token>
<expression> ::= <token> | <token> <expression>
```

Operation definitions:
`not` – binary negation operation (example: `not 101 = 010`)
`shr` – binary right shift operation (example: `shr 101 = 10`)
`shl` – binary left shift operation (example: `shl 101 = 1010`)
`xor` – binary exclusive or operation (example: `0111 xor 1011 = 1100`)
`and` – binary and operation (example: `0111 and 1011 = 0011`)
`or` – binary or operation (example: `0111 or 1011 = 1111`)

All unary operators have higher priority than binary ones and binary operators must be calculated from left to right (their priority is considered equal). Before any unary operation all additional leading zeros should be removed (for example `0011` becomes `11` and `000` becomes `0`), and before a binary one you should align binary numbers by adding leading zeros if necessary (for example `11 xor 101` becomes `011 xor 101`).

INPUT

The number of tests `T` ($T \leq 100$) is given on the first line. Each of next `T` lines contains an expression itself. Length of the expression will be always less than 1000 characters.

OUTPUT

For each test case output a single line `"Case T: N"`. Where `T` is the test case number (starting from 1) and `N` is the value of evaluated expression in the same binary form. Answer should not contain any leading zeroes.

SAMPLE INPUT

```
4
shl not 101
not 11 and 111
111 xor not 0
shl 0
```

SAMPLE OUTPUT

```
Case 1: 100
Case 2: 0
Case 3: 110
Case 4: 0
```

E — Binomial Theorem

Time Limit: 1 sec
Memory Limit: 32 MB

John likes mathematics a lot. His main passion is the binomial theorem. However it is rather hard to calculate binomial coefficients, so he decided to write a computer program that can expand any power of a sum into a sum of powers. Mathematically it can be written like this:
$(a+b)^k = x_1 a^k + x_2 a^{k-1} b + x_3 a^{k-2} b^2 + \ldots + x_{k+1} b^k$
where $x_{1\ldots k+1}$ are binomial coefficients $x_i = C_k^i$.

INPUT

There is a number of tests T ($T \leq 100$) on the first line. After `T` test follows. Each test is written on a single line in form of `(a+b)^k`. Where `a` and `b` are same variables names. Variables names are strings constructed from '`a`'–'`z`' characters. And `k` ($1 \leq k \leq 50$) is a power that you need to raise the sum. You can assume that there are no lines longer than 100 characters.

OUTPUT

For each test output a single line "`Case N: T`". Where `N` is the test number (starting from 1) and `T` is an expanded expression (see examples for clarification). By the way, you shouldn't output coefficients and powers equal to one.

SAMPLE INPUT

```
3
(a+b)^1
(alpha+omega)^2
(acm+icpc)^3
```

SAMPLE OUTPUT

```
Case 1: a+b
Case 2: alpha^2+2*alpha*omega+omega^2
Case 3: acm^3+3*acm^2*icpc+3*acm*icpc^2+icpc^3
```

F — Brainfuck

Time Limit: 3 sec
Memory Limit: 32 MB

Recently your friend Bob has bought a new brainfuck programmable LED display. However executing a program directly on LED display takes huge amount of time. Because of this, now Bob decided to write the interpreter of display instruction set in order to test and debug all his programs on his PC and only after that execute his code on LED display. However Bob knows only one programming language — its certainly brainfuck (otherwise he would not have bought this LED display). So he asks you to write him an interpreter.

A display's program is a sequence of commands executed sequentially. The commands of the display processor is a subset of brainfuck language commands. The commands that processor was capable to execute were '>', '<', '+', '–' and '.', which are described in the table below. Moreover, this LED display has an array of 100 bytes of circular memory (initialised with zeros) and a pointer to this array (initialised to point to the leftmost byte of the array). This means, that after incrementing a pointer, which points to the rightmost byte of memory, it will point to the leftmost byte and vice versa. Individual bytes are circular as well, so increasing a 255 gives a 0 and vice versa.

Command	**Description**
>	Increment the pointer (to point to the next cell to the right).
<	Decrement the pointer (to point to the next cell to the left).
+	Increment (increase by one) the byte at the pointer.
–	Decrement (decrease by one) the byte at the pointer.
.	Output the value of the byte at the pointer.

INPUT

There is a number of tests `T` ($T \leq 100$) on the first line. After `T` tests follow. Each test case is a sequence of display processor commands on a separate line. You can assume that line length is less than 100000.

OUTPUT

For each test output a single line "`Case T: D`". Where `T` is the test number (starting from 1) and `D` is display's memory dump in hexademical numeration system after executing given brainfuck program. Every byte must be separated exactly by one space character. See examples for clarification. Please note, that the sample input and output is divided into several lines only for convenience.

SAMPLE INPUT

```
1
..++<><<+++>>++++++++++++++++++++++++++>>>+++
<+...++<><<+++>>++++++++++++++++++++++++++>>>
+++<+...++<><<+++>>++++++++++++++++++++++++++
>>>+++<+.
```

SAMPLE OUTPUT

```
Case 1: 1F 00 20 03 1D 03 01 03 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 03 00
```

G — Checkers

Time Limit: 1 sec
Memory Limit: 32 MB

In this problem you are given a (n, n) checkerboard with many black checker pieces and only one white. You have to calculate the number on different paths for the white piece to become a king (reach the top board row). You can assume that only white piece is moving, all black pieces are always standing still on their positions. All checkers moving rules (for the white piece) are applied:

- Target cell must be free.
- From position $(x; y)$ checker can move only forward by one of diagonals to position $(x + 1; y + 1)$ or $(x - 1; y + 1)$.
- If cell $(x + 1; y + 1)$ or $(x - 1; y + 1)$ is occupied by enemy piece it can jump over to cell $(x + 2; y + 2)$ or $(x - 2; y + 2)$ respectively.

INPUT

The number of tests `T` ($T \leq 100$) is given on the first line. Each test starts with integer `N` ($1 \leq N \leq 100$) the board size. Next, `N` lines follow with `N` characters each describing the board itself. On the board `W` stands for white piece, `B` for black one and `.` for unoccupied cell. There will be only one white piece in each test.

OUTPUT

For each test case output a single line `"Case T: S"`. Where `T` is the test case number (starting from 1) and `S` number of paths for the white piece to become a king. As this number can be huge, output it modulo `1000007`.

SAMPLE INPUT

```
2
4
....
....
....
..W.
8
.B.B.B..
........
........
..B.....
........
..B.....
.W......
........
```

SAMPLE OUTPUT

```
Case 1: 5
Case 2: 1
```

H — Coming Home

Time Limit: 1 sec
Memory Limit: 32 MB

John is going back home after a party. Currently he is standing on a bus station and waiting for a bus to arrive. There is a timetable of arriving buses near the station. Beside this, John also knows the amount of time that is needed to travel with specific bus. As he has only one ticket, there are no possibilities to change the bus somewhere in the middle of a trip in order to make it shorter. Can you help John to calculate minimal time that he needs to get home?

INPUT

There is a number of tests `T` ($T \leq 100$) on the first line. Each test case contains the number of buses `K` ($1 \leq K \leq 100$) and current time (in format `HH:MM`). Each of the next K lines contain arrival time of the bus (in the same format as current time) and travelling time $0 \leq Q \leq 1000$ needed for John to get home (in minutes). Refer to the sample input as an example.

OUTPUT

For each test case output a single line `"Case T: N"`. Where `T` is the test case number (starting from 1) and `N` minimal time (in minutes) needed for John to go back home.

SAMPLE INPUT

```
2
1 18:00
19:30 30
2 18:00
19:00 100
20:00 30
```

SAMPLE OUTPUT

```
Case 1: 120
Case 2: 150
```

I — Dice

Time Limit: 1 sec
Memory Limit: 32 MB

Mary add Sue are playing with dices. Rules are simple: at the begging each of them puts coin on the table and roll a dice. Wins a player who rolled a larger number. If numbers are the same, coins stay on the table for a next round. In order to make this game more interesting they decided to play now with normal dices, but with dice that can have arbitrary number of bones, from 0 till 9. However each round must be played with same dice by both players.

Girls have been playing this game for a day long, till Mary run out of coins (nevertheless she had more coins at the beginning of the game). Now Mary is confused. How could she have lost all her coins? She thinks that Sue had been cheating. Before each roll Mary wrote on a paper numbers of bones on each side of the dice. Now she wonders if same dice was always used during one round. Help her to find it out.

INPUT

On the first line there is the total number of test cases T ($T \leq 10^3$), next T lines follows. Each line contains two six digit numbers, each digit stands for number of bones on side of a dice in this order: top, bottom, front, left, back, right.

OUTPUT

For each test case output line `"Equal"` if two dices are equals, or `"Not Equal"` otherwise.

SAMPLE INPUT

```
3
345678 345678
123123 123456
123456 351624
```

SAMPLE OUTPUT

```

Equal
Not Equal
Equal
```

J — Divisor Game

Time Limit: 2 sec
Memory Limit: 32 MB

Steve is playing a game with numbers. He picks up a random positive number N and finds the largest positive number not bigger than N that has the most divisors. As N becomes larger it's more and more difficult for Steve to avoid mistakes when counting the divisors and he asks you to write a program. You argue that it is a very easy task to just find the divisors and suggest that you could solve the original task of Steve as well.

INPUT

You are given a number of tests T ($T \leq 50000$). Each test on a single line specifies a number N ($1 \leq N \leq 10^6$).

OUTPUT

You need to find the largest number not bigger than N that has the most divisors. For each test output one line containing the answer to the game.

SAMPLE INPUT

```
3
1
10
37
```

SAMPLE OUTPUT

```

1
10
36
```

K — DNA

Time Limit: 1 sec
Memory Limit: 32 MB

Alan was comleting his internship at chemistry faculty. At the beginning internship seemed to be easy. However one day dean of the faculty of chemistry have given John a certain task to solve. Alan was asked to perform DNA and protein modelling using free OpenEye software kit.

As Alan like all programmers is very lazy, he would like to use computer as much as possible. So now he needs to generate all possible DNA mutations and share them to OpenEye software. Alan decided to use computer not only for OpenEye software but also for mutation generation purpose.

At the beginning of the internship Alan studying a little bit of chemistry science, so he knows that DNA consisting of 4 elements (**A**denine, **G**uanine, **C**ytosine, and **T**hymine) and can be described as a sequence of four letters (for example: `GATCC`). The K-th mutation of inital DNA sequence of length N is called a sequence that can be produced by replacing ((possibly to the same nucleotide)) exactly K elements of the sequence (for example `GAT` is the 1-st mutation of `GGT` and the 2-nd mutation of `TTT`).

Alan is given initial DNA sequence and maximal power of its possible mutataion. Can you produce all possible mutated DNA sequences for Alan?

INPUT

The number of tests `T` ($T \leq 50$) is given on the first line. Each test case if described by two lines: one the first is number `N` ($N \leq 10$) and `K` ($K \leq 5$), on the second line is written DNA sequence of length `N`.

OUTPUT

For each test case print `M` — the number of different DNA mutations. After this print all mutated DNA sequences in alphabetical order. Refer to the sample output for details.

SAMPLE INPUT

```
1
3 1
AAA
```

SAMPLE OUTPUT

```
10
AAA
AAC
AAG
AAT
ACA
AGA
ATA
CAA
GAA
TAA
```

L — DNA II

Time Limit: 1 sec
Memory Limit: 32 MB

As it was mention in the previous task, any DNA sequence consists from four bases: adenine (A), cytosine (C), guanine (G) and thymine (T) and can be written as a string constructed from characters A, C, G or T. As any strings, DNA sequences can be ordered alphabetically. In order to reduce amount of memory DNA sequence can be described by its length and index in the ordered (index starts from 0) set of all possible DNA sequences of some specific length. So you are asked to write a program that will encode any given DNA sequence in a pair of numbers.

Lets take for example all DNA sequences of length 2. They form the ordered set:
{ AA; AC; AG; AT; CA; CC; CG; CT; GA; GC; GG; GT; TA; TC; TG; TT }
So sequence "CC" can be described by the pair (2;5) (2 is a length and 5 is an index in the set), "AG" by (2,2), "TG" by (2,14) and so on.

INPUT

The number of tests `T` ($T \leq 100$) is given on the first line. Each of next `T` lines contains DNA sequence `S` of maximal length of 30 characters.

OUTPUT

For each test case output a single line `"Case T: (A;B)"`. Where `T` is the test case number (starting from 1) and `(A:B)` is a pair describing the DNA sequence by a given approach.

SAMPLE INPUT

```
3
AC
ATA
TAGCAGCAGCAGCGAA
```

SAMPLE OUTPUT

```

Case 1: (2:1)
Case 2: (3:12)
Case 3: (16:3374617184)
```

M — Emigration

Time Limit: 2 sec
Memory Limit: 32 MB

Lithuanian migration department has announced that approximately 90 thousand Lithuanians will have emigrated by the end of 2010. This is many times more than any other European country and if no measures will be taken the whole population will be subject to extinction.

The department is investigating possible ways of attracting Lithuanians back to their country, and most importantly reduce existing emigration. The next step of the investigation includes consolidation of data received from Lithuanian embassies across the world, in order to identify the regions where Lithuanians have migrated: United Kingdom, Norway, Ireland among others.

To identify what measures are best the following poll was arranged in some of the countries. People where asked to choose what government activity can bring them home. Each activity i has associated cost X in litas. Total cost to implement the activity consists of fixed and variable costs. Fixed cost is the same for all countries. Variable cost depends on the number of emigrants in that particular country. You also know that activity i has efficiency E, given in percent, meaning that $E\%$ emigrants will be attracted back home. Efficiency can be different for different countries. Each activity is always implemented to all people in a country, but only some will return back home to Lithuania.

Given that no effective activity compensating emigration has been found so far the government has decided to diversify the activities and implement each activity at most in 1 country, to mitigate the risk of missing potentially undervalued activities. However, different activities can be implemented in different countries. You are asked to find the number of Lithuanians that can be attracted back home according to the poll results.

INPUT

On the first line of input you are given a number of tests T ($T \leq 100$). Each test starts with a line containing 3 positive integers separated by a space character: K ($K \leq 5$) — number of activities, N ($N \leq 10$) — number of countries, M ($M \leq 1000$) — money that government can spend on activities in millions of litas. Then data about activities follow containing K lines. Each line describes an activity contains 1 upper-case latin alphabet letter (activity name) and 2 positive integers: C_p — fixed cost in thousands of litas, C_v variable cost per person in litas. First line of data for each country specifies the number of activities $K_i \leq K$ applicable to that country, and $P_i (P_i \leq 10^6)$ — the number of emigrants in that country. K_i lines follow each specifying the activity index, and an integer number specifying efficiency of the activity in percent of country's population. When calculating number of people attracted by activity in a country round to the nearest integer, and if needed half up.

OUTPUT

For each test case print a line containing the maximum number of Lithuanians that can be attracted back home.

SAMPLE INPUT

```
1
2 1 100
A 1000 2
B 500 1
2 1000000
A 50%
B 25%
```

SAMPLE OUTPUT

```
500000
```

N — Equation

Time Limit: 3 sec
Memory Limit: 32 MB

Your friend John was told to find an ingeter solution to the equation:
$x_1 + 2x_2 + 4x_3 + 8x_4 = 15 \textit{ where } x_i \geq 0$
John spent all week thinking and could find only one solution, it is $x_1 = 1; x_2 = 1; x_3 = 1; x_4 = 1$. However, he thinks that is not the only possible integer solution to this equation, but he isn't sure about it. You have decided to help your friend to prove or deny his assumption. In order to do this you should write a program that finds the number of integer solutions for a given equation (let us leave the fun of searching solutions to John). On the other hand it is rather boring to write program for this particular equation, so you generalize it into the form:
$x_1 + 2x_2 + 4x_3 + 8x_4 + 16x_5 + ... + 2^t x_t = K \textit{ where } x_i \geq 0$.
Can you manage this?

INPUT

The number of tests `T` ($T \leq 10^5$) is given on the first line. Each of next `T` lines contains two numbers `K M` ($0 \leq K \leq 10^5; 1 \leq M \leq 10^{15}$). Where `K` is the number from equation and `M` is modulo. `M` can be always factorized into primes numbers not larger than `150`.

OUTPUT

For each test case output a single line `"Case T: N"`. Where `T` is the test case number (starting from 1) and `N` is the number of different integer vectors that are solutions to a given equation. As number `N` can be very large, output it modulo `M`.

SAMPLE INPUT

```
3
15 99
101 123
1234 536870912
```

SAMPLE OUTPUT

```
Case 1: 26
Case 2: 111
Case 3: 176223474
```

O — Extra Spaces

Time Limit: 1 sec
Memory Limit: 32 MB

In programming multiple whitespaces are used to only to make code more readable, so mostly all programming languages totally ignore multiple spaces in code (except for some esoteric ones). In general there are different types of whitespace characters: space itself, tabs, newline symbol, various control characters, etc. Tabs and spaces bring one or the biggest holywar to a programmers world as there is no common rule what to use for code indentation – tab or space characters.

In this holywar you stand for tab side and your project code convention requires to use only them for code indentation. However you have recently spotted that someone is using space characters instead of it. Four spaces and tab character look the same in our text editor, so you have decided to write a parser that will change all consequent space characters to one. After that you would be able to determine amount of corrupted code.

INPUT

The number of tests `T` ($T \leq 100$) is given on the first line. At first line of each test there is integer `N` ($N \leq 50$). Next `N` lines with text that must be processed for extra spaces follow. Maximal line length is equal to 500 characters.

OUTPUT

For each test case output a single line `"Case T:"`. Where `T` is the test case number (starting from 1). Next `N` lines must be output with input text having no consequent spaces. By the way, always leave a blank line between tests. Please refer to the sample output for clarity.

SAMPLE INPUT

```
2
3
Sample test one:
  there was 2 spaces and
here are also  2  spaces
2
Sample test two:
     there was 4 spaces
```

SAMPLE OUTPUT

```
Case 1:
Sample test one:
 there was 2 spaces and
here are also 2 spaces

Case 2:
Sample test two:
 there was 4 spaces
```

P — Galactic Bonding

Time Limit: 2 sec
Memory Limit: 32 MB

Calvin likes to lie in a field and look at the night sky. Since he doesn't know any real star constellations, he makes them up: if two stars are close to each other, they must belong to the same constellation. He wants to name them all, but fears to run out of names. Can you help him and count how many constellations there are in the sky?

Two stars belong to the same constellation if distance between their projections on a two-dimensional sky plane isn't more than D units.

INPUT

There is a number of tests T ($T \leq 50$) on the first line. Each test case contains the number of stars N ($0 \leq N \leq 1000$) a real distance D ($0.00 \leq D \leq 1000.00$). Next N lines have a pair of real coordinates XY ($-1000.00 \leq X, Y \leq 1000.00$) for each star. Real numbers in the input will have at most 2 digits after a decimal point.

OUTPUT

For each test case output a single line `"Case T: N"`. Where T is the test case number (starting from 1) and N is the number of constellations.

SAMPLE INPUT

```
2
5 1.5
1.0 0.1
2.0 0.0
5.0 0.2
6.0 0.4
3.0 -0.1
3 4.0
121.12 254.06
645.04 301.85
912.49 568.96
```

SAMPLE OUTPUT

```
Case 1: 2
Case 2: 3
```

Q — Hic-Hac-Hoe

Time Limit: 2 sec
Memory Limit: 32 MB

Everybody knows how to play tic-tac-toe. If accidentally you do not know the rules of this game, you can always consult Wikipedia.

In this problem we will use a slightly different version of tic-tac-toe. First, the game board is not limited to 3x3 cells, but considered infinite. Also in order to win a player must get not 3 but at least k noughts or crosses in a line (horizontal, vertical or diagonal).

In modified tic-tac-to version it is not so easy to determine a winner. So in this problem you will be given a list of turns performed by the players during the game and you need to determine the winner.

INPUT

There is a number of tests `T` ($T \leq 100$) on the first line. Each test case is described by the two numbers `n k` ($n \leq 10^5, k \leq 5$), where `n` stands for number of turns for both players and `k` for winning line size. Next line contains `n` pairs of signed 32-bit integers `x y` — coordinates of each players turn. All turns have been performed sequentially by both players and crosses have always started a game.

OUTPUT

For each test case output a single line `"Case T: S"`. Where `T` is the test case number (starting from 1) and `S` is equal to `"crosses"` or `"noughts"` if one of them has a winning line. If nobody yet has won a game output `"none"` and if both players have winning lines output `"error"` for `"S"`.

SAMPLE INPUT

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

SAMPLE OUTPUT

```
Case 1: crosses
Case 2: error
```

R — In The Airport

Time Limit: 1 sec
Memory Limit: 32 MB

It's a Wednesday night and Bob is sitting at the airport and waiting for an airplain to Lithuania. The challenge he is facing is to come up with an easy task for the qualification contest on Saturday in KTU. Unfortunately, he doesn't have any more ideas because it's already late and he is too exhausted.

To freshen up, he thinks that he should get a drink and a piece of cake. Waitress gives Bob a menu to choose from. As he was too busy thinking about how tired he is, he simply decided to go for a drink and piece of cake which are the closest to the average of all items on the menu. If he needs to choose, he always goes for the cheaper item. Unfortunately waitress is not very good at math, and obviously she needs your help!

INPUT

You are given ($T \leq 100$) tests on the first line. Each test starts with a line containing the integers N, M, K, where ($2 \leq N \leq 1000$) is the number of items on the menu, ($1 \leq M \leq N$) is the number of cakes on the menu and ($1 \leq K \leq N$) is the number of drinks on the menu. Next line contains exactly N positive integers, where the first M integers specify the prices for cakes, next K integers ($0 \leq P_i \leq 10^9$) specify drink prices, and the following prices are for all other items on the menu.

OUTPUT

For each test case you have to output a line `"Case #T: A B"`. Where T is the test case number, A specifies the closest price of the cake to the average of all items, and B is the closest price of the drink to the average of all items. Refer sample output for details.

SAMPLE INPUT

```
2
4 1 1
1 2 3 4
5 1 2
500000000 5 6 1000000000 1000000000
```

SAMPLE OUTPUT

```
Case #1: 1 2
Case #2: 500000000 6
```

S — LAMAbpo

Time Limit: 10 sec
Memory Limit: 32 MB

LAMAbpo is a Lithuanian organization which decides whether to admit the student to university or not. In this task your are required to write an application for a part of the LAMAbpo system. Given the score for each of the study programmes included in the application of high-school graduates you have to find the results of the admissions.

Essentially, the application includes: the list of study programmes ordered by priority from highest to lowest (application entry) and the personal data of a graduate. The contest is performed as indicated in the rules below:

- First, the highest priority study programme is considered. If the contestant's score is enough, then he is admitted to this course and other courses are not considered.
- If the score is not enough the next study programme in the application is considered.
- The algorithm proceeds until the application entry is found where the contestant meets the requirements (i.e. the score is big enough) or all entries are marked as considered.

Be aware, that if you mark an entry as considered you must be sure that either a contestant may not be accepted to the study programme or he must be accepted to this study programme.

INPUT

You are given a single test. The test starts with a number of study programmes `S` and a number of high-school graduates `M`. `S` and `M` are separated by a single space. Then a list of study programmes follows. Each study programme description consists of 2 lines:

- Title of a study programme (a string with at most 200 ASCII characters)
- A string with 2 numbers separated by a single space — programme identification code `I` ($1 \leq I \leq 1000$) and a number of students to admit study programme `A` ($0 \leq A \leq 35000$), respectively.

There may not be 2 programmes with the same identification code in the input file and total number of entries of all study programmes will not exceed `E` ($E \leq 200000$).
The description of the study programmes is followed by `M` applications. Each application consits of the following lines:

- First name and last name of a high-school graduate (at most 100 characters)
- The number of study programmes `L` indicated in the application by the graduate and a unique positive integer `U` ($U \leq 10^9$) given to a contestant (which is considered if 2 contestants have the same score) separated by a space.
- `L` ($L \leq 16$) lines containing the identification code of a study programme and the contestant score `G` ($G \leq 10000$) with at most 2 digits after decimal point. The numbers are separated by a single space. Score may not be negative. The study programme may not appear in the same application more than once. There will be no identification code with the study programme that does not exist.

OUTPUT

You have to output a list of study programmes in order as they appear in the input file with the following information:

- Name of the study programme on a single line
- Number of admitted contestants (this number may be less than the number of available places)

- A list of contestants admitted to the study programme. Each line in this list consists of the first and last name of a contestant. List must be ordered by contestants' score with highest first. If 2 contestans have the same score then a contestant with a highest number `U` should be first. Each name is preceded by a number indicating a place of the graduate in the list and a space character (refer sample output).

SAMPLE INPUT

```
2 4
Informatika
620 2
Matematika
621 1
Simas Simaitis
2 9529032
620 10.5
621 10.0
Jonas Jonaitis
1 3984329
621 11.5
Erika Erikaite
2 2499345
621 11.5
620 11.5
Vaclovas Vaclovaitis
2 1451044
621 9.5
620 20.5
```

SAMPLE OUTPUT

```
Informatika
2
1. Vaclovas Vaclovaitis
2. Erika Erikaite
Matematika
1
1. Jonas Jonaitis
```

T — Lucky Numbers

Time Limit: 1 sec
Memory Limit: 32 MB

Every person has its own numbers that he considers lucky. Usually the numbers are fixed like 3 or 7 and do not depend on anything. This lucky number model seems to be very primitive for John, so he decided to upgrade it for his own use. Maybe more complex model will bring more luck to him?

John has added a dependency for lucky numbers on specific integer `N` (for example `N` can be ordinal number of day in year or some other meaning). For each `N` John considers some number `X` lucky if and only if fraction $\frac{X}{\sqrt{N-X}}$ value is integer and greater than zero.

INPUT

The number of tests `T` ($T \leq 100$) is given on the first line. `T` lines follow, each of them contains one integer `N` ($1 \leq N \leq 10^9$) described above.

OUTPUT

For each test case output a single line `"Case T: S"`. Where `T` is the test case number (starting from 1) and `S` is increasing sequence of numbers considered lucky by John for specified `N`. Please refer to the sample output for clarity.

SAMPLE INPUT

```
3
16
109
33
```

SAMPLE OUTPUT

```
Case 1: 12 15
Case 2: 108
Case 3: 24 32
```

U — Polygon

Time Limit: 1 sec
Memory Limit: 32 MB

John has been given a segment of lenght N, however he needs a polygon. In order to create a polygon he has cut given segment K times at random positions (uniformly distributed cuts). Now he has $K + 1$ much shorter segments. What is the probability that he can assemble a polygon using all new segments?

INPUT

The number of tests `T` ($T \leq 1000$) is given on the first line. `T` lines follow, each of them contains two integers `N K` ($1 \leq N \leq 10^6; 1 \leq K \leq 50$) described above.

OUTPUT

For each test case output a single line `"Case #T: F"`. Where `T` is the test case number (starting from 1) and `F` is the result as simple fraction in form of `N/D`. Please refer to the sample output for clarity.

SAMPLE INPUT

```
2
1 1
2 2
```

SAMPLE OUTPUT

```
Case #1: 0/1
Case #2: 1/4
```

V — Round Trip

Time Limit: 1 sec
Memory Limit: 32 MB

John recently has had his birthday party. As John likes traveling, all invited guests cooperated and decided to buy various tickets as a present. Each ticket can be used only once to travel from some city A to city B or vice versa. As guests were cooperating, they decided that no tickets will be between the same cities.

Now John has huge amount of tickets and needs to plan his trip. But first, he wishes to know the cities that now he can possibly visit and come back home. In other words you have to find all cities John can visit by making a round trip from the home city through that particular city.

INPUT

The number of tests `T` ($T \leq 100$) is given on the first line. Each test starts with 3 integers `N M C` ($N \leq 100; M \leq 1000; 1 \leq C \leq N$). Where `N` stands for number of cities (cities are numbered from `1` to `N`) and `M` is number of tickets John has. `C` describes his living city number. Next `M` lines describe tickets with 2 integers `X Y` ($1 \leq T \leq 100$). `X` and `Y` are cities on the ticket.

OUTPUT

For each test case output a single line `"Case T: S"`. Where `T` is the test case number (starting from 1) and `S` is single space separated list of cities that John could possibly visit. This list must be sorted in increasing order. If John can't find a single city to visit print `"none"` intead (i.e. `Case 2: none`).

SAMPLE INPUT

```
2
6 7 1
1 2
1 3
2 6
4 6
2 5
5 3
4 2
2 1 2
1 2
```

SAMPLE OUTPUT

```
Case 1: 2 3 4 5 6
Case 2: none
```

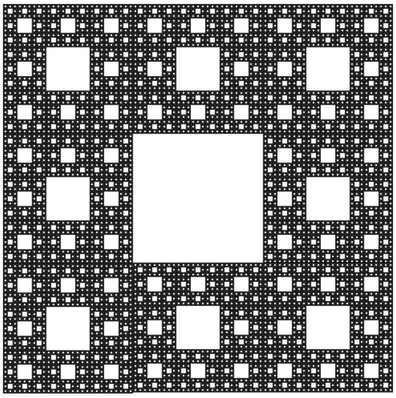
W — Sierpinski Carpet

Time Limit: 1 sec
Memory Limit: 32 MB

The Sierpinski carpet is a typical plane fractal. The construction of it begins with a square. The square is cut into 9 congruent subsquares in a 3-by-3 grid, and the central subsquare is removed. The same procedure is then applied recursively to the remaining 8 subsquares, ad infinitum.

Let's call the first figure (single square) from which we begin carpet construction S_0, next figure (combined 8 squares) S_1, S_2 figure is combined of 64 squares and so on.

Here is an example of full Sierpinski carpet (S_∞):



In this problem you will be given a point in the plain and you have to find the maximal figure S_N to which this point still belongs.

INPUT

The number of tests `T` ($T \leq 100$) is given on the first line. Each of next `T` lines contains two floating point numbers cordinates `X` ($0 < X < 1$) and `Y` ($0 < Y < 1$) of a given point. Point's cordinates are given with maximal precision of 6 digits after decimal point.

OUTPUT

For each test case output a single line `"Case T: N"`. Where `T` is the test case number (starting from 1) and `N` is the maximal index of figure S that point still belongs to. If point is in Sierpinski Carpet itself (S_∞) then `N` must be equal to `-1`.

SAMPLE INPUT

```
2
0.111 0.111
0.123 0.123
```

SAMPLE OUTPUT

```
Case 1: 10
Case 2: 1
```

X — Switch The Lights

Time Limit: 5 sec
Memory Limit: 32 MB

Kaunas University of Technology has bought a new light toggling system from one of the cheapest manufacturers in China. It consists of N lamps and M switches. Each switch has a subset of lights assigned to it, and when toggled, it changes the state of all the lights in the subset from on to off and vice versa. Also the system contains the main switch which is used to turn turn off all lights.

The authorities installed the switches at different locations in the univerisity. But one day the main switch went down. Now they are not able to turn off all lights by using the main switch. Unfortunately noone understands the Chineese documentation of the system, so we must wait for support from manufacturers. But we have good programmers, and we are interested in finding the minimal number of switches required to turn off all lights in the university. Initially, all lights are turned on.

INPUT

The first line of input contains the number of tests T ($T \leq 50$). Each test case is a set of lines. First line of each test case contains 2 positive integers N ($N \leq 15$) and M ($M \leq 100$) separated by a space character. Next M lines contain N integers K ($K_i \in \{1, 0\}$) separated by a space character (if the i-th integer is 1 then the i-th light is toggled by the switch).

OUTPUT

For each test case output a single line "Case T: N". Where T is the test case number (starting from 1) and N is the minimal number of switches required. If it is impossible to turn off all lights N should be equal to "IMPOSSIBLE".

SAMPLE INPUT

```
2
2 2
0 1
1 0
3 2
1 0 1
1 1 0
```

SAMPLE OUTPUT

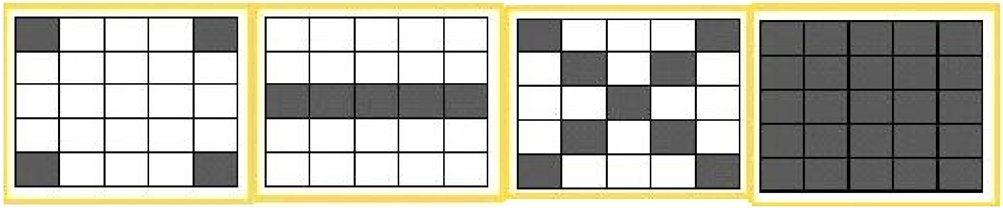
```
Case 1: 2
Case 2: IMPOSSIBLE
```

Y — Tele-loto

Time Limit: 1 sec
Memory Limit: 32 MB

Tele-loto is a lottery game that is held every weekend in Lithuania. Most of the population know this game and almost everyone have played it at least once. In this task you are asked to write an application which finds the winnings given N balls, the lottery tickets and amount of money for each combination. Possible combinations are listed below:

- Corners (must be filled within the first 35 balls)
- MidLine (must be filled within the first 40 balls)
- Diagonals (must be filled within the first 45 balls)
- Table (until one of the players wins the game)



Totally 75 balls are in the game each with a unique number from 1 to 75. The i-th column in the ticket contains unique numbers in the interval $[(i-1)*15+1; i*15]$, no other numbers may appear in the column. If the ticket wins more than one combination the total money won is equal to the sum of money won for each combination. When at least one of the players fills the whole table, the game is over and no more balls are drawn. Pay attention that combination MidLine is applied only to a single middle line!

INPUT

The first line contains the number of tests `T` ($T \leq 100$). Then `T` tests follow. On the first line of each test there are 2 numbers: `N` ($0 \leq N \leq 75$) — number of balls, `L` ($L \leq 10^3$) — number of tickets. On the second line there are `N` different integers indicating the values of the drawn lucky balls separated by a single space character. Next line contains 4 integers `V` ($V_i \leq 1000$) indicating the values of each combination in left-to-right order as in the picture. Then 5 lines follow for each ticket with 25 different integers indicating the numbers on the ticket in the order as in the picture. Each line contains exactly 5 integers separated by a single space character.

OUTPUT

For each test case output the line `"Case T:"` where `T` is a test number starting from 1. Then `L` lines follow each indicating the amount of money the ticket won in the order they appear in the input. The test cases must be separated by a blank line. Refer sample output for details.

SAMPLE INPUT

```
1
9 1
12 67 8 75 4 30 42 54 74
2 5 10 1000
12 20 36 57 67
2 28 45 59 63
4 30 42 54 74
5 26 34 49 70
8 16 37 48 75
```

SAMPLE OUTPUT

```
Case 1:
7
```

Z — Time Deposit

Time Limit: 1 sec
Memory Limit: 32 MB

In order to reduce the effect of inflation you have recently put all your savings into a bank for the time deposit. According to rules of this deposit, interest is calculated only on working days. And now you are interested in knowing the real value of your deposit (with interest). But in order to calculate the value of the deposit you have to know the number of working days. So you decided to write a program which is able to calculate the number of working days in any given period of time.

Normally a working day is any day except Saturday and Sunday, however there are exceptions. For each exceptional date you will be given if it is a working or not.

INPUT

There is a number of tests `T` ($T \leq 100$) on the first line. Each test starts with the integer `N` ($N \leq 100$), the number of exception dates. On next `N` lines there are dates in format `yyyy-mm-dd` followed by a character `W` or `H`. `W` stands for the exceptional working day and `H` — for holiday. Next two lines contain period start and end dates respectively. All dates will be between `1975-01-01` and `2025-12-31` and there won't be more than one exceptional date on each date in single test.

OUTPUT

For each test case output a single line `"Case T: N"`. Where `T` is the test case number (starting from 1) and `N` is the number of working days between start and end dates inclusive.

SAMPLE INPUT

```
2
3
2008-01-01 H
2008-01-02 H
2008-01-11 W
2008-01-01
2008-02-01
0
2008-01-01
2009-01-01
```

SAMPLE OUTPUT

```
Case 1: 22
Case 2: 263
```

HINT

For this task you may assume that all leap years are divisible by 4.