Practice Round APAC test 2016

A. Bad Horse

B. Captain Hammer

C. Moist

Questions asked

Problem A. Bad Horse

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

Small input 1 12 points	Solve A-small-1
Small input 2 21 points	Solve A-small-2

Problem

As the leader of the Evil League of Evil, Bad Horse has a lot of problems to deal with. Most recently, there have been far too many arguments and far too much backstabbing in the League, so much so that Bad Horse has decided to split the league into two departments in order to separate troublesome members. Being the Thoroughbred of Sin, Bad Horse isn't about to spend his valuable time figuring out how to split the League members by himself. That what he's got you -- his loyal henchman -- for.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a positive integer **M** on a line by itself -- the number of troublesome pairs of League members. The next **M** lines each contain a pair of names, separated by a single space.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is either "Yes" or "No", depending on whether the League members mentioned in the input can be split into two groups with neither of the groups containing a troublesome pair.

Limits

$1 \leq \mathbf{T} \leq 100$.
Each member name will consist of only letters and the underscore character.
Names are case-sensitive.
No pair will appear more than once in the same test case.
Each pair will contain two distinct League members.

Small dataset

$1 \leq \mathbf{M} \leq 10$.

Large dataset

$1 \leq \mathbf{M} \leq 100$.

Sample

Input	Output
2 1 Dead_Bowie Fake_Thomas_Jefferson 3 Dead_Bowie Fake_Thomas_Jefferson Fake_Thomas_Jefferson Fury_Leika Fury_Leika Dead_Bowie	Case #1: Yes Case #2: No

Practice Round APAC test 2016

A. Bad Horse

B. Captain Hammer

C. Moist

Questions asked

Problem B. Captain Hammer

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

Small input
22 points

Solve B-small

Problem

The Hamjet is a true marvel of aircraft engineering. It is a jet airplane with a single engine so powerful that it burns all of its fuel instantly during takeoff. The Hamjet doesn't have any wings because who needs them when the fuselage is made of a special Wonderflonium isotope that makes it impervious to harm.

Piloting the Hamjet is a not a job for your typical, meek-bodied superhero. That's why the Hamjet belongs to Captain Hammer, who is himself impervious to harm. The G-forces that the pilot endures when taking a trip in the Hamjet are legen-dary.

The Hamjet takes off at an angle of θ degrees up and a speed of **V** meters per second. **V** is a fixed value that is determined by the awesome power of the Hamjet engine and the capacity of its fuel tank. The destination is **D** meters away. Your job is to program the Hamjet's computer to calculate θ given **V** and **D**.

Fortunately, the Hamjet's Wondeflonium hull is impervious to air friction. Even more fortunately, the Hamjet doesn't fly too far or too high, so you can assume that the Earth is flat, and that the acceleration due to gravity is a constant 9.8 m/s^2 down.

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each line will contain two positive integers -- **V** and **D**.

Output

For each test case, output one line containing "Case #x: θ", where x is the case number (starting from 1) and θ is in degrees up from the the horizontal. If there are several possible answers, output the smallest positive one.

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

Limits

$1 \le \mathbf{T} \le 4500$;
$1 \le \mathbf{V} \le 300$;
$1 \le \mathbf{D} \le 10000$;
It is guaranteed that each test case will be solvable.

Sample

Input	Output
3 98 980 98 490 299 1234	Case #1: 45.0000000 Case #2: 15.0000000 Case #3: 3.8870928

Problem C. Moist

Small input 1 4 points	Solve C-small-1
Small input 2 6 points	Solve C-small-2

Problem

Moist has a hobby -- collecting figure skating trading cards. His card collection has been growing, and it is now too large to keep in one disorganized pile. Moist needs to sort the cards in alphabetical order, so that he can find the cards that he wants on short notice whenever it is necessary.

The problem is -- Moist can't actually pick up the cards because they keep sliding out his hands, and the sweat causes permanent damage. Some of the cards are rather expensive, mind you. To facilitate the sorting, Moist has convinced Dr. Horrible to build him a sorting robot. However, in his rather horrible style, Dr. Horrible has decided to make the sorting robot charge Moist a fee of $1 whenever it has to move a trading card during the sorting process.

Moist has figured out that the robot's sorting mechanism is very primitive. It scans the deck of cards from top to bottom. Whenever it finds a card that is lexicographically smaller than the previous card, it moves that card to its correct place in the stack above. This operation costs $1, and the robot resumes scanning down towards the bottom of the deck, moving cards one by one until the entire deck is sorted in lexicographical order from top to bottom.

As wet luck would have it, Moist is almost broke, but keeping his trading cards in order is the only remaining joy in his miserable life. He needs to know how much it would cost him to use the robot to sort his deck of cards.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each one starts with a line containing a single integer, **N**. The next **N** lines each contain the name of a figure skater, in order from the top of the deck to the bottom.

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 number of dollars it would cost Moist to use the robot to sort his deck of trading cards.

Limits

$1 \leq \mathbf{T} \leq 100$.
Each name will consist of only letters and the space character.
Each name will contain at most 100 characters.
No name with start or end with a space.
No name will appear more than once in the same test case.
Lexicographically, the space character comes first, then come the upper case letters, then the lower case letters.

Small dataset

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

Large dataset

$1 \leq \mathbf{N} \leq 100$.

Sample

Input	Output
2 2 Oksana Baiul Michelle Kwan 3 Elvis Stojko Evgeni Plushenko Kristi Yamaguchi	Case #1: 1 Case #2: 0

Round A APAC Test 2016

A. Googol String

B. gCube

C. gCampus

D. gSnake

Questions asked

Submissions

Googol String

7pt | Not attempted **2083/5209 users** correct (40%)

12pt | Not attempted **957/1730 users** correct (55%)

gCube

8pt | Not attempted **1557/2234 users** correct (70%)

16pt | Not attempted **855/1488 users** correct (57%)

gCampus

10pt | Not attempted **493/1232 users** correct (40%)

15pt | Not attempted **227/482 users** correct (47%)

gSnake

13pt | Not attempted **121/629 users** correct (19%)

19pt | Not attempted **41/88 users** correct (47%)

Top Scores

cebrusfs	100
sgtlaugh	100
usaxena95	100
akovski	100
NAFIS	100
liuyibo1994	100
dtyfc	100
Legendks	100
Shaon	100
jki14	100

Problem A. Googol String

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

Small input 7 points	Solve A-small
Large input 12 points	Solve A-large

Problem

A "0/1 string" is a string in which every character is either `0` or `1`. There are two operations that can be performed on a 0/1 string:

- **switch**: Every `0` becomes `1` and every `1` becomes `0`. For example, "100" becomes "011".
- **reverse**: The string is reversed. For example, "100" becomes "001".

Consider this infinite sequence of 0/1 strings:

S_0 = ""

S_1 = "0"

S_2 = "001"

S_3 = "0010011"

S_4 = "001001100011011"

...

$S_N = S_{N-1}$ + "0" + **switch**(**reverse**(S_{N-1})).

You need to figure out the Kth character of S_{googol}, where googol = 10^{100}.

Input

The first line of the input gives the number of test cases, **T**. Each of the next **T** lines contains a number **K**.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the **K**th character of S_{googol}.

Limits

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

Small dataset

$1 \le \mathbf{K} \le 10^5$.

Large dataset

$1 \le \mathbf{K} \le 10^{18}$.

Sample

Input	Output
4 1 2 3 10	Case #1: 0 Case #2: 0 Case #3: 1 Case #4: 0

 Google Home - Terms and Conditions - Privacy Policies and Principles

Powered by

Google Cloud Platform

Problem B. gCube

Small input 8 points	Solve B-small
Large input 16 points	Solve B-large

Problem

Googlers are very interested in cubes, but they are bored with normal three-dimensional cubes and also want to think about other kinds of cubes! A "D-dimensional cube" has D dimensions, all of equal length. (D may be any positive integer; for example, a 1-dimensional cube is a line segment, and a 2-dimensional cube is a square, and a 4-dimensional cube is a hypercube.) A "D-dimensional cuboid" has D dimensions, but they might not all have the same lengths.

Suppose we have an **N**-dimensional cuboid. The **N** dimensions are numbered in order (0, 1, 2, ..., N - 1), and each dimension has a certain length. We want to solve many subproblems of this type:

1. Take all consecutive dimensions between the $\mathbf{L_i}$-th dimension and $\mathbf{R_i}$-th dimension, inclusive.

2. Use those dimensions to form a D-dimensional cuboid, where D = $\mathbf{R_i}$ - $\mathbf{L_i}$ + 1. (For example, if $\mathbf{L_i}$ = 3 and $\mathbf{R_i}$ = 6, we would form a 4-dimensional cuboid using the 3rd, 4th, 5th, and 6th dimensions of our **N**-dimensional cuboid.)

3. Reshape it into a D-dimensional cube **that has exactly the same volume as that D-dimensional cuboid**, and find the edge length of that cube.

Each test case will have **M** subproblems like this, all of which use the same original **N**-dimensional cuboid.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.

Each test case begins with two integers **N** and **M**; **N** is the number of dimensions and **M** is the number of queries. Then there is one line with **N** positive integers $\mathbf{a_i}$, which are the lengths of the dimensions, in order. Then, **M** lines follow. In the ith line, there are two integers $\mathbf{L_i}$ and $\mathbf{R_i}$, which give the range of dimensions to use for the ith subproblem.

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). After that, output **M** lines, where the ith line has the edge length for the ith subproblem. An edge length will be considered correct if it is within an absolute error of 10^{-6} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{a_i} \le 10^9$.
$0 \le \mathbf{L_i} \le \mathbf{R_i} < \mathbf{N}$.

Small dataset

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

Large dataset

$1 \le \mathbf{N} \le 1000$.
$1 \le \mathbf{M} \le 100$.

Sample

Input	Output
2 2 2 1 4 0 0	Case #1: 1.000000000 2.000000000 Case #2:

```
0 1     1.414213562
3 2     2.449489743
1 2 3
0 1
1 2
```

Problem C. gCampus

Small input 10 points	Solve C-small
Large input 15 points	Solve C-large

Problem

Company G has a main campus with **N** offices (numbered from 0 to **N** - 1) and **M** bidirectional roads (numbered from 0 to **M** - 1). The ith road connects a pair of offices (U_i, V_i), and it takes C_i minutes to travel on it (in either direction).

A path between two offices X and Y is a series of one or more roads that starts at X and ends at Y. The time taken to travel a path is the sum of the times needed to travel each of the roads that make up the path. (It's guaranteed that there is at least one path connecting any two offices.)

Company G specializes in efficient transport solutions, but the CEO has just realized that, embarrassingly enough, its own road network may be suboptimal! She wants to know which roads in the campus are *inefficient*. A road is inefficient if and only if it is not included in **any** shortest paths between **any** offices.

Given the graph of offices and roads, can you help the CEO find all of the inefficient roads?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each case begins with one line with two integers **N** and **M**, indicating the number of offices and roads. This is followed by **M** lines containing three integers each: $\mathbf{U_i}$, $\mathbf{V_i}$ and $\mathbf{C_i}$, indicating the ith road is between office $\mathbf{U_i}$ and office $\mathbf{V_i}$, and it takes $\mathbf{C_i}$ minutes to travel on it.

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). Then output the road numbers of all of the inefficient roads, in increasing order, each on its own line. (Note that road 0 refers to the first road listed in a test case, road 1 refers to the second road, etc.)

Limits

$0 < \mathbf{C_i} \leq 1000000$.

Small dataset

$1 \leq \mathbf{T} \leq 10$.
$1 \leq \mathbf{N} = \mathbf{M} \leq 100$.

Large dataset

$1 \leq \mathbf{T} \leq 3$.
$1 \leq \mathbf{N} \leq 100$.
$1 \leq \mathbf{M} \leq 10000$.

Sample

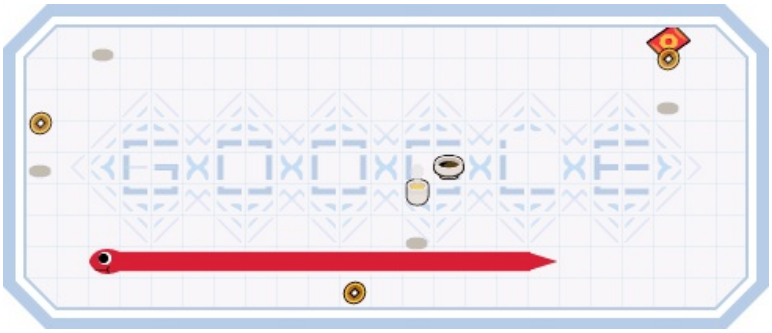
Input	Output
2 3 3 0 1 10 1 2 3 2 0 3 3 3 0 1 10 1 2 3 2 1 3	Case #1: 0 Case #2:

Problem D. gSnake

Small input 13 points	Solve D-small
Large input 19 points	Solve D-large

Problem

Alex is a huge fan of the Snake game.



Note: This Google Doodle does not exactly match the rules of the Snake game we will describe below. It is only intended to give you a general idea of what the game looks like.

Alex just learned how to program and wants to develop his own version of Snake, with the following rules:

- The game board has **R** rows and **C** columns. The top left cell of the board has coordinates (1, 1), and the bottom right cell has coordinates (**R**, **C**).
- At the start of the game, in every cell with coordinates (r, c) such that r + c is odd, there is one piece of food. No other cells have food.
- The snake's body is always an ordered, connected sequence of one or more cells on the board. The first cell of the sequence is called the "head" of the snake. The second cell (if any) shares an edge (not just a corner) with the first cell, and so on. The last cell in the sequence is called the "tail" of the snake.
- The snake's head is always facing in one of four directions: left, up, right, or down.
- At the start of the game, the snake is at cell (1, 1) and has a length of one (that is, the snake consists of only a head), and the head faces right.
- At each integer time (1 second, 2 seconds, etc.), the head of the snake will move into the adjacent cell that its head is facing toward. The board is **cyclic**, i.e., trying to move off an edge will cause the head to appear on the opposite edge of the board. For example, if the snake is at (1, **C**) and its head is facing right, the head will next move to (1, 1). If the snake is at (1, **C**) and its head is facing up, the head will next move to (**R**, **C**).
- When the snake's head moves into a cell with no food, the snake does not grow. The snake's second cell (if any) moves to the place where the snake's head was, the snake's third cell (if any) moves to the place where the second cell was, and so on.
- When the snake's head moves into a cell with a piece of food, it eats the food (meaning that cell no longer has food), and grows its body. A new head is created in the cell where the food was. The cell that was the snake's head becomes the snake's second cell, the cell that was the snake's second cell (if any) becomes the snake's third cell, and so on.
- If, after a move is complete, the snake's head is in the same place as one of another of its cells, the snake dies and the game ends immediately. (Note that if the snake's head moves toward a cell where its tail was, the game will not end, because the tail will move out of the way before the move is complete.)
- In the game, the player can let the snake perform some turn actions. Each action A_i will happen between the T_ith and T_i+1 th seconds. There are two possible actions: "L" and "R". An "L" action will turn the head 90 degrees to the left, so, for example, if the snake had been facing down before, it would face right after. An "R" action will turn the head 90 degrees to the right, so, for example, if the snake had been facing down before, it would face left after.
- The game has a time limit: it will end after the move on the $\mathbf{10^9}$th second is complete (if the game has even gone on that long!)

To test the game, Alex has written a series of TURN actions. Your task is to simulate that series of actions, and tell Alex the final length of the snake when the game is over. Remember that the game can end either because the snake's

head and another cell of its body are in the same place after a move is complete, or because time runs out. In the former case, you should count both the head and the overlapping cell of its body as two separate cells, for the purpose of determining length.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test cases starts with three integers **S**, **R**, and **C**, where **S** gives the number of turn actions and **R** and **C** represent the number of rows and columns of the board. **S** lines follow; the ith of these lines has an integer X_i, then a character A_i that is either L or R. Each of these lines corresponds to performing an action between X_ith and X_i+1 th seconds. It's guaranteed that the actions are given in time order and there will never be more than one action between the same two seconds. However, you should note that the game may end before the snake gets to execute all of these actions.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the length of the snake when the game is over.

Limits

$1 \leq \mathbf{T} \leq 10$.

Small dataset

$1 \leq \mathbf{R}, \mathbf{C} \leq 100$;
$1 \leq \mathbf{S} \leq 100$;
$1 \leq X_i \leq 2000$.

Large dataset

$1 \leq \mathbf{R}, \mathbf{C} \leq 100000$;
$1 \leq \mathbf{S} \leq 100000$;
$1 \leq X_i \leq 1000000$.

Sample

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

Round B APAC Test 2016

A. Travel

B. gWheels

C. gNumbers

D. Albocede DNA

Questions asked

Problem A. Travel

Small input 6 points	Solve A-small
Large input 12 points	Solve A-large

Problem

There are **N** cities in Chelsea's state (numbered starting from 1, which is Chelsea's city), and **M** bidirectional roads directly connect them. (A pair of cities may even be directly connected by more than one road.) Because of changes in traffic patterns, it may take different amounts of time to use a road at different times of day, depending on when the journey starts. (However, the direction traveled on the road does not matter -- traffic is always equally bad in both directions!) All trips on a road start (and end) exactly on the hour, and a trip on one road can be started instantaneously after finishing a trip on another road.

Chelsea loves to travel and is deciding where to go for her winter holiday trip. She wonders how quickly she can get from her city to various other destination cities, depending on what time she leaves her city. (Her route to her destination may include other intermediate cities on the way.) Can you answer all of her questions?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.

The first line of each test case contains three integers: the number **N** of cities, the number **M** of roads, and the number **K** of Chelsea's questions.

2**M** lines -- **M** pairs of two lines -- follow. In each pair, the first line contains two different integers x and y that describe one bidirectional road between the x-th city and the y-th city. The second line contains 24 integers Cost[t] ($0 \le t \le 23$) that indicate the time cost, in hours, to use the road when departing at t o'clock on that road. It is guaranteed that $Cost[t] \le Cost[t+1]+1$ ($0 \le t \le 22$) and $Cost[23] \le Cost[0]+1$.

Then, an additional **K** lines follow. Each contains two integers **D** and **S** that comprise a question: what is the fewest number of hours it will take to get from city 1 to city **D**, if Chelsea departs city 1 at **S** o'clock?

Output

For each test case, output one line containing "Case #x: ", where x is the case number (starting from 1), followed by **K** distinct space-separated integers that are the answers to the questions, in order. If Chelsea cannot reach the destination city for a question, no matter which roads she takes, then output -1 for that question.

Limits

$1 \le x, y \le \mathbf{N}$.
1 ≤ all Cost values ≤ 50.
$1 \le \mathbf{D} \le \mathbf{N}$.
$0 \le \mathbf{S} \le 23$.

Small dataset

$1 \le \mathbf{T} \le 100$.
$2 \le \mathbf{N} \le 20$.
$1 \le \mathbf{M} \le 100$.
$1 \le \mathbf{K} \le 100$.

Large dataset

$1 \le \mathbf{T} \le 5$.
$2 \le \mathbf{N} \le 500$.
$1 \le \mathbf{M} \le 2000$.
$1 \le \mathbf{K} \le 5000$.

Sample

Input

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

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
2 3
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 1
3 3
3 1 2
1 2
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2
3 4
3 3 3
1 2
7 23 23 25 26 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12
1 3
10 11 15 26 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 1
2 3
7 29 28 27 26 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12
2 14
3 3
3 21
```

Output

```
Case #1: 1 2
Case #2: 1 -1
Case #3: 17 26 13
```

Problem B. gWheels

Small input 5 points	Solve B-small
Large input 14 points	Solve B-large

Problem

A typical mountain bike has two groups of gears: one group connected to the pedals, and one group connected to the rear tire. A gear group consists of many gears, which usually have different numbers of teeth. A chain connects one of the gears in the pedal gear group to one of the gears in the tire gear group, and this determines the ratio between the cyclist's pedaling speed and the tire speed. For example, if the chain connects a gear with 5 teeth on the pedals to a gear with 10 teeth on the tires, the ratio will be 1/2, since the cyclist needs to make the pedal gear rotate twice to make the tire rotate once. The cyclist can change the chain to connect any one gear from the pedal group to any one gear from the tire group.

You have just bought a special new mountain bike with *three* groups of gears: one connected to the pedals, one connected to the tire, and one extra group in between. This mountain bike has two chains; the first chain must always connect one gear from the pedal gear group to one gear on the extra gear group, and the second chain must always connect one gear from the extra gear group to one gear on the tire gear group. Moreover, the two chains cannot both use the same gear from the extra gear group.

Given the numbers of teeth on the available gears on the pedals, extra, and tire groups, is it possible to make the ratio (between pedaling speed and tire speed) be exactly **P**/**Q**? For a given set of gears, you may need to answer multiple such questions.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each begins with one line with 3 integers $\mathbf{N_p}$, $\mathbf{N_e}$, and $\mathbf{N_t}$, representing the numbers of gears on the pedals, extra, and tire groups. Then, three more lines follow. These contain $\mathbf{N_p}$, $\mathbf{N_e}$, and $\mathbf{N_t}$ integers, respectively, representing the numbers of teeth on the different gears on the pedals, extra, and tire gear groups, respectively. (It is guaranteed that the numbers of teeth on the gears within a group are all distinct.) The next line after that consists of one integer, **M**, the number of questions. Finally, there are **M** lines, each with 2 integers, **P** and **Q**, representing the target ratio. (It is not guaranteed that this ratio is a reduced fraction.)

Output

For each test case, first output one line containing "Case #x:", where x is the test case number (starting from 1). Then output one line for each question within the test case, in the order that the questions were presented: Yes if it's possible to make the ratio **P**/**Q**, and No if it's impossible.

Limits

$1 \le \mathbf{T} \le 100$.
1 ≤ the number of teeth on each gear ≤ 10000.
1 ≤ all values of **P**, **Q** ≤ 10^9.
$1 \le \mathbf{M} \le 10$.

Small dataset

$1 \le \mathbf{N_p}, \mathbf{N_t} \le 10$.
$2 \le \mathbf{N_e} \le 10$.

Large dataset

$1 \le \mathbf{N_p}, \mathbf{N_t} \le 2000$.
$2 \le \mathbf{N_e} \le 2000$.

Sample

Input	Output
1	Case #1:
1 2 3 5	No Yes

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

For the first question in the test case, it's impossible to get the ratio 1/1 since this would require both chains to be on the same gear in the extra gear group, which is not allowed.

For the second question in the test case, you can make the first chain connect the 5-tooth gear on the pedal gear group to the 4-tooth gear on the extra gear group, and make the second chain connect the 6-tooth gear on the extra gear group to the 3-tooth gear on the tire gear group. With this setup, the ratio is 5/2.

Problem C. gNumbers

Small input
8 points

Solve C-small

Large input
16 points

Solve C-large

gNumbers

Googlers are crazy about numbers and games, especially number games! Two Googlers, Laurence and Seymour, have invented a new two-player game based on "gNumbers". A number is a gNumber if and only if the sum of the number's digits has no positive divisors other than 1 and itself. (In particular, note that 1 is a gNumber.)

The game works as follows: First, someone who is not playing the game chooses a starting number **N**. Then, the two players take turns. On a player's turn, the player checks whether the current number C is a gNumber. If it is, the player loses the game immediately. Otherwise, the player chooses a prime factor P of C, and keeps dividing C by P until P is no longer a factor of C. (For example, if the current number were 72, the player could either choose 2 and repeatedly divide by 2 until reaching 9, or choose 3 and repeatedly divide by 3 until reaching 8.) Then the result of the division becomes the new current number, and the other player's turn begins.

Laurence always gets to go first, and he hates to lose. Given a number **N**, he wants you to tell him which player is certain to win, assuming that both players play optimally.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of a starting number **N**.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the winner's name: either `Laurence` or `Seymour`.

Limits

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

Small dataset

$1 < \mathbf{N} \leq 1000$.

Large dataset

$1 < \mathbf{N} \leq 10^{15}$.

Sample

Input	Output
9 2 3 4 6 8 9 30 36300 1000000000000000	Case #1: Seymour Case #2: Seymour Case #3: Laurence Case #4: Laurence Case #5: Laurence Case #6: Laurence Case #7: Seymour Case #8: Laurence Case #9: Seymour

In Case #1, 2 is already a gNumber, since the sum of its digits is 2, which has no positive divisors other than 1 and itself. So Laurence immediately loses, which means Seymour wins. The same is true for Case #2.

In Case #3, 4 is not a gNumber, since the sum of its digits is 4, which has a positive divisor other than 1 and itself (namely, 2). 4 has one prime factor (2), so Laurence must choose this factor and repeatedly divide 4 by it, which leaves him with 1. Then, Seymour begins his turn with 1, which is a gNumber. So he

loses and Laurence wins.

Problem D. Albocede DNA

Small input 16 points	Solve D-small
Large input 23 points	Solve D-large

The DNA of the Albocede alien species is made up of 4 types of nucleotides: a, b, c, and d. Different Albocedes may have different sequences of these nucleotides, but any Albocede's DNA sequence obeys all of the following rules:

- It contains at least one copy of each of a, b, c, and d.
- All as come before all bs, which come before all cs, which come before all ds.
- There are exactly as many `'a's` as `'c's`.
- There are exactly as many `'b's` as `'d's`.

For example, abcd and aabbbccddd are valid Albocede DNA sequences. acbd, abc, and abbccd are not.

The Albocede-n is an evolved species of Albocede. The DNA sequence of an Albocede-n consists of one or more valid Albocede DNA sequences, concatenated together end-to-end. For example, abcd and aaabcccdaabbbccdddabcd are valid Albocede-n DNA sequences. (Observe that a valid Albocede-n DNA sequence is not necessarily also a valid Albocede DNA sequence.)

From one of your alien expeditions, you retrieved an interesting sequence of DNA made up of only as, bs, cs, and ds. You are interested in how many of the different subsequences of that sequence would be valid Albocede-n DNA sequences. (Even if multiple different selections of nucleotides from the sequence produce the same valid subsequence, they still all count as distinct subsequences.) Since the result may be very large, please find it modulo 1000000007 ($10^9 + 7$).

Input

The first line of the input gives the number of test cases, **T**. Each of the next **T** lines contains a string **S** that consists only of the characters a, b, c, and d.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the output of the x^{th} test case.

Limits

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

Small dataset

$1 \leq$ **length of S** ≤ 50.

Large dataset

$1 \leq$ **length of S** ≤ 500.

Sample

Input	Output
5 abcd aaaabcd aaaabbccd abcdabcdaabccd b	Case #1: 1 Case #2: 4 Case #3: 28 Case #4: 71 Case #5: 0

 Google Home - Terms and Conditions - Privacy Policies and Principles

Powered by

Google Cloud Platform

Round C APAC Test 2016

A. gRanks
B. gFiles
C. gGames
D. gMatrix

Questions asked 2

Submissions

gRanks
6pt Not attempted 1263/2254 users correct (56%)
10pt Not attempted 923/1243 users correct (74%)

gFiles
9pt Not attempted 529/1189 users correct (44%)
17pt Not attempted 222/493 users correct (45%)

gGames
10pt Not attempted 85/221 users correct (38%)
18pt Not attempted 13/44 users correct (30%)

gMatrix
11pt Not attempted 826/1065 users correct (78%)
19pt Not attempted 157/549 users correct (29%)

Top Scores

johngs	100
cchao	100
NAFIS	100
exprosic	100
orenguy	83
nhho	82
yaray	82
BananaTree	82
mkrjn99	82
tapasjain	82

Problem A. gRanks

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

Small input 6 points	Solve A-small
Large input 10 points	Solve A-large

Problem

There are many great athletes in the world, and it's very hard to say who is the best in the world at a particular sport, especially when different athletes have won different competitions. Here's one possible system for ranking athletes:

1. Determine the number **P** of finishing places in any competition that will be worth points for athletes, and how many points $\mathbf{S_i}$ each of those finishing places is worth. For example, for **P** = 3, one possible assignment would be 1000 points for 1st place, 500 for 2nd place, and 300 for 3rd place, and 0 for anything below that. (We assume there are no ties within competitions.)

2. Since not all competitions are equally important, assign a weight $\mathbf{W_i}$ to each one. The score gained by an athlete for a competition will be the points from step 1, modified by the weight for that competition. For example, we may decide that Olympics has a weight of 5, and, continuing with our example from above, the winner of the Olympics would receive 5 * 1000 = 5000 points.

3. Since we don't want to reward athletes simply for participating in many competitions, we count only the **M** highest scores received by an athlete across all competitions. For example, if **M** = 2 and an athlete earns scores of 1000*5, 500*1, and 300*3 in three different competitions, only the 5000 and 900 would be counted.

You are given the points per finishing place, the weights of the competitions, and the results of the competitions. Can you rank all of the athletes who appeared in the competitions? If multiple athletes have the same score, they will share the same rank and listed in alphabetical order of their names.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each test case consists of the following:

1. One line with an integer **P**, the number of top places for which points are awarded.
2. One line consists with **P** integers representing the scores $\mathbf{S_i}$ for the top places, starting with first place and continuing down the places.
3. One line with an integer **N**, the number of competitions. 4. **N** lines, each of which represents a competition. Each line begins with $\mathbf{W_i}$, the weight of this competition, and continues with the **P** names of the athletes who finished in the top **P** places. They are listed in descending order starting from first place.
5. One line with an integer **M**, the maximum number of competitions counted toward an athlete's score.

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). Then output one line for each athlete, from highest rank to lowest rank, with the format `r: name`, where `r` is the rank of the athlete and `name` is the name of the athlete. You need to rank all of the athletes that appeared in the input.

Limits

$1 \le \mathbf{T} \le 10$.
$1 \le \mathbf{S_i} \le 1000$.
$\mathbf{S_i} > \mathbf{S_{i+1}}$.
$1 \le \mathbf{W_i} \le 1000$.
Each name consists only of characters A through Z, and is at most 10 characters long.

Small dataset

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

Large dataset

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

$1 \leq \mathbf{M} \leq 100$.

Sample

Input	Output
1 2 1000 500 6 5 BOLT GAY 4 GAY BOLT 1 GAY TIANBING 1 GAY PEIMENG 1 TIANBING LARRY 1 PEIMENG LARRY 2	Case #1: 1: BOLT 2: GAY 3: PEIMENG 3: TIANBING 5: LARRY

In the first case, Bolt scored a total of 7000 in his two competitions. Gay would have scored a total of 8500 if all competitions were counted, but since only the top 2 competitions are counted in this case, Gay scored 6500 and ranked second. Since Peimeng and Tianbing both scored 1500, they both ranked 3rd and listed by their names. Larry is last, since he scored only 1000.

Problem B. gFiles

Small input 9 points	Solve B-small
Large input 17 points	Solve B-large

Problem

Alien tech company G has a very old file transfer tool that is still in use today. While the tool is running, it reassures users by giving status updates on both the percentage of files transferred so far and the number of files transferred so far. The status updates during the process might look like this:

```
 20% |==>-------|     1 files transferred
100% |==========|     5 files transferred
```

But the percentage isn't precise; it is simply truncated before the decimal point (i.e. floored to the next lowest or equal 1%). That is, both 1.2% and 1.7% would be displayed as 1%.

Some users may want to know the exact total number of files, so you want to modify the tool so that the status updates look like this:

```
 20% |==>-------|     1 out of 5 files transferred
100% |==========|     5 out of 5 files transferred
```

But you've realized that it may or may not be possible to know the number of files. Given the status updates that the tool displays, either figure out how many files there are, or determine that it can't be done (i.e., there are multiple possible values for the number of files). The status updates are not guaranteed to occur at regular intervals and are not guaranteed to occur whenever a file is transferred.

Input

The first line contains **T**, the number of test cases. **T** test cases follow. Each test case consists of one line with an integer **N**, the number of status updates output by the tool, followed by **N** lines with the format P_i K_i, where P_i and K_i are integers representing the percentage and number of files transferred at some point in the process. The updates are given listed in chronological order -- that is, both the P_i values and the K_i values are nondecreasing throughout a test case.

Output

For each case, output a line starts with "Case #x: y", where x is the number of the case (starting from 1), and y is either the total number of files, or -1 if that number is ambiguous.

Limits

$1 \le \mathbf{T} \le 50$.
$1 \le \mathbf{N} \le 100$.
$0 \le P_i \le 100$

Small dataset

$0 \le K_i \le 2000$
The answer is guaranteed not to exceed 2000.

Large dataset

$0 \le K_i \le 10^{15}$
The answer is guaranteed not to exceed 10^{15}.

Sample

Input	Output
3 2 20 1	Case #1: 5 Case #2: -1 Case #3: 23

```
100 5
10
25 241
27 262
43 407
44 413
57 536
64 601
67 637
84 789
95 893
96 903
10
0 0
8 2
8 2
17 4
30 7
39 9
69 16
73 17
82 19
91 21
```

Round C APAC Test 2016

A. gRanks
B. gFiles
C. gGames
D. gMatrix

Questions asked 2

Submissions

gRanks
6pt | Not attempted **1263/2254 users** correct (56%)
10pt | Not attempted **923/1243 users** correct (74%)

gFiles
9pt | Not attempted **529/1189 users** correct (44%)
17pt | Not attempted **222/493 users** correct (45%)

gGames
10pt | Not attempted **85/221 users** correct (38%)
18pt | Not attempted **13/44 users** correct (30%)

gMatrix
11pt | Not attempted **826/1065 users** correct (78%)
19pt | Not attempted **157/549 users** correct (29%)

Top Scores

johngs	100
cchao	100
NAFIS	100
exprosic	100
orenguy	83
nhho	82
yaray	82
BananaTree	82
mkrjn99	82
tapasjain	82

Problem C. gGames

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

Small input 10 points	Solve C-small
Large input 18 points	Solve C-large

Problem

The country of elves is planning to hold an elimination tournament, and there are $2^{\mathbf{N}}$ elves who would like to take part. At the start of the tournament, they will be given unique ID numbers from 1 to $2^{\mathbf{N}}$, and the Elf President will line them up in some order.

The tournament is a series of matches between two elves, and every match has one winner and one loser (there are no ties). In the first round, the first elf in the line competes against the second elf in the line, the third elf competes against the fourth elf, and so on. After the first round, the $2^{\mathbf{N-1}}$ elves who lost leave the line, and the $2^{\mathbf{N-1}}$ elves who won remain where they are. Then, the remaining elves play the second round in the same way: the first remaining elf in the line competes against the second remaining elf in the line, the third remaining elf competes against the fourth remaining elf, and so on. After **N** rounds, there will be only one elf remaining, and that elf is the winner.

M of the elves are sensitive, which means that they will be very sad if they have to compete in matches against their friends during the games. Specifically, the ith elf will be sad if they have to compete with their friends in the first $\mathbf{K_i}$ rounds. (Note that friendship is not necessarily mutual: if one elf considers another elf to be a friend, the other elf does not necessarily consider that elf to be a friend.)

The Elf President wants to know: is there a way to specify the initial positions of all $2^{\mathbf{N}}$ elves to guarantee that no elf will be sad, no matter what happens in the tournament?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with two integers **N** and **M**, then **M** sets of two lines each, in which the first line has integers $\mathbf{E_i}$, $\mathbf{K_i}$, and $\mathbf{B_i}$ for one elf, and the second has $\mathbf{B_i}$ integer ID numbers of that elf's friends.

Output

For each test case, output one line containing "Case #x: ", where x is the case number (starting from 1), followed by YES or NO.

Limits

$1 \le \mathbf{T} \le 200$.
$0 \le \mathbf{M} \le 2^{\mathbf{N}}$.
$1 \le \mathbf{E_i} \le 2^{\mathbf{N}}$.
$1 \le \mathbf{K_i} \le \mathbf{N}$.
$\mathbf{M} \le \text{sum}(\mathbf{B_i}) \le \min(2 * \mathbf{M}, 2^{\mathbf{N}})$.

Small dataset

$1 \le \mathbf{N} \le 3$.

Large dataset

$\mathbf{N} = 4$.

Sample

Input	Output
3 1 1 1 1 1 2 2 2 1 1 1 2 3 1 1 4	Case #1: NO Case #2: YES Case #3: YES

```
3 3
1 2 2
3 4
2 2 2
5 6
7 1 1
8
```

Round C APAC Test 2016

A. gRanks
B. gFiles
C. gGames
D. gMatrix

Questions asked 2

Problem D. gMatrix

Problem

You have a square **N** by **N** matrix M of nonnegative integers. We would like to make a list of the maximum values in every sub-matrix of size **K** by **K** within M, and then find the sum of those values together. (Note that the same entry of M might be the maximum value in more than one sub-matrix, in which case it will show up multiple times in the list.) Can you find that sum?

To simplify the input of the matrix, you are given two arrays **A** and **B** of length **N**, and two integers **C** and **X**. Then the entry M_{ij} (for the ith row and jth column of the matrix) equals ($\mathbf{A_i}$*i+$\mathbf{B_j}$*j + **C**) mod **X**, where i and j are in the range [1, **N**].

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case begins with one line with four integers, **N**, **K**, **C** and **X**. Then there are two lines with **N** integers each, representing the arrays **A** and **B**.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the sum of the maximum values in all sub-matrices of size **K** by **K**.

Limits

1 ≤ **T** ≤ 100.
1 ≤ $\mathbf{A_i}$, $\mathbf{B_i}$ ≤ 100000.
1 ≤ **C** ≤ 100000.
1 ≤ **X** ≤ 1000000007.
1 ≤ **K** ≤ **N**.

Small dataset

1 ≤ **N** ≤ 50.

Large dataset

1 ≤ **N** ≤ 3000.

Sample

Input	Output
3 1 1 1 5 1 1 2 1 5 11 1 2 3 4 3 2 3 109 6 4 3 2 1 5	Case #1: 3 Case #2: 19 Case #3: 80

In the first test case, the matrix is:
3
So the sum of maximum values is 3.

In the second test case, the matrix is:
9 3
1 6
So the sum of maximum values is 19.

In the third test case, the matrix is:
11 11 24

```
13 13 26
14 14 27
```

Questions asked 1

Submissions

Dynamic Grid

6pt	Not attempted **1392/1881 users** correct (74%)
8pt	Not attempted **1288/1368 users** correct (94%)

gBalloon

9pt	Not attempted **353/666 users** correct (53%)
17pt	Not attempted **266/338 users** correct (79%)

IP Address Summarization

10pt	Not attempted **123/236 users** correct (52%)
19pt	Not attempted **73/118 users** correct (62%)

Virtual Rabbit

11pt	Not attempted **18/166 users** correct (11%)
20pt	Not attempted **3/8 users** correct (38%)

Top Scores

nhho	100
sundar95	80
Shaon	80
ajkrish95	80
ojas.deshpande	80
NAFIS	69
JunoYu	69
wcwswsws	69
karanaggarwal	69
VotBear	69

Problem A. Dynamic Grid

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

Small input 6 points	Solve A-small
Large input 8 points	Solve A-large

Problem

We have a grid with **R** rows and **C** columns in which every entry is either 0 or 1. We are going to perform **N** operations on the grid, each of which is one of the following:

- Operation M: Change a number in one cell of the grid to 0 or 1
- Operation Q: Determine the number of different *connected regions* of 1s. A connected region of 1s is a subset of cells that are all 1, in which any cell in the region can be reached from any other cell in the region by traveling between cells along edges (**not** corners).

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with one line with two integers, **R** and **C**, which represent the number of rows and columns in the grid. Then, there are **R** lines of **C** characters each, in which every character is either `0` or `1`. These lines represent the initial state of the grid.

The next line has one integer, **N**, the number of operations to perform on the grid. **N** more lines follow; each has one operation. All operation Ms will be of the form `M x y z`, meaning that the cell at row x and column y should be changed to the value z. All operation Qs will be of the form `Q`.

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). Then, for every operation Q in the test case, in order, output one line containing the number of connected regions of 1s.

Limits

$1 \le \mathbf{T} \le 10$.
$1 \le \mathbf{R}, \mathbf{C} \le 100$.
$0 \le \mathbf{x} < \mathbf{R}$.
$0 \le \mathbf{y} < \mathbf{C}$.
$0 \le \mathbf{z} \le 1$.

Small dataset

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

Large dataset

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

Sample

Input	Output
1 4 4 0101 0010 0100 1111 7 Q M 0 2 1 Q M 2 2 0 Q M 2 1 0 Q	Case #1: 4 2 2 2

Round D APAC Test 2016

A. Dynamic Grid
B. gBalloon
C. IP Address Summarization
D. Virtual Rabbit

Questions asked 1

Submissions

Dynamic Grid
6pt | Not attempted 1392/1881 users correct (74%)
8pt | Not attempted 1288/1368 users correct (94%)

gBalloon
9pt | Not attempted 353/666 users correct (53%)
17pt | Not attempted 266/338 users correct (79%)

IP Address Summarization
10pt | Not attempted 123/236 users correct (52%)
19pt | Not attempted 73/118 users correct (62%)

Virtual Rabbit
11pt | Not attempted 18/166 users correct (11%)
20pt | Not attempted 3/8 users correct (38%)

Top Scores

nhho	100
sundar95	80
Shaon	80
ajkrish95	80
ojas.deshpande	80
NAFIS	69
JunoYu	69
wcwswswws	69
karanaggarwal	69
VotBear	69

Problem B. gBalloon

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

Small input 9 points	Solve B-small
Large input 17 points	Solve B-large

Problem

The G tech company has deployed many balloons. Sometimes, they need to be collected for maintenance at the company's tower, which is located at horizontal position 0. Each balloon is currently at horizontal position $\mathbf{P_i}$ and height $\mathbf{H_i}$.

G engineers can move a balloon up and down by sending radio signals to tell it to drop ballast or let out air. But they can't move the balloon horizontally; they have to rely on existing winds to do that.

There are **M** different heights where the balloons could be. The winds at different heights may blow in different directions and at different velocities. Specifically, at height j, the wind has velocity $\mathbf{V_j}$, with positive velocities meaning that the wind blows left to right, and negative velocities meaning that the wind blows right to left. A balloon at position P at a height with wind velocity V will be at position P+V after one time unit, P+2V after two time units, etc. If a balloon touches the tower, it is immediately collected.

It costs $|\ H_{original} - H_{new}\ |$ points of energy to move one balloon between two different heights. (This transfer takes no time.) You have **Q** points of energy to spend, although you do not need to spend all of it. What is the least amount of time it will take to collect all the balloons, if you spend energy optimally?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follows. The first line of each case has three integers **N**, **M**, and **Q**, representing the number of balloons, the number of height levels, and the amount of energy available.
The second line has **M** integers; the jth value on this line (counting starting from 0) is the wind velocity at height j.
Then, **N** more lines follow. The ith of these lines consists of two integers, $\mathbf{P_i}$ and $\mathbf{H_i}$, representing the position and height of the ith balloon.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the minimum number of time units needed to collect all of the balloons, returns IMPOSSIBLE if it's impossible to collect all the balloons using the energy given.

Limits

Small dataset

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{N} \le 10$.
$1 \le \mathbf{M} \le 10$.
$-10 \le \mathbf{V_j} \le 10$.
$1 \le \mathbf{Q} \le 10$.
$0 \le \mathbf{H_i} < \mathbf{M}$.
$-10 \le \mathbf{P_i} \le 10$.

Large dataset

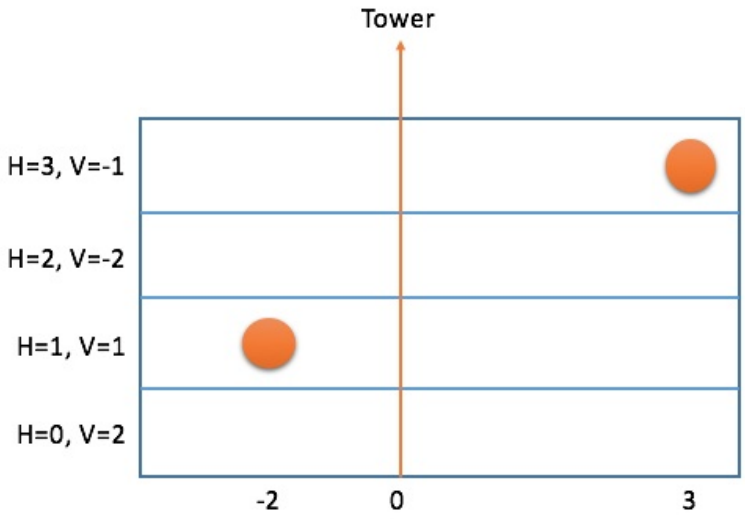
$1 \le \mathbf{T} \le 25$.
$1 \le \mathbf{N} \le 100$.
$1 \le \mathbf{M} \le 1000$.
$-100 \le \mathbf{V_j} \le 100$.
$1 \le \mathbf{Q} \le 10000$
$0 \le \mathbf{H_i} < \mathbf{M}$.
$-10000 \le \mathbf{P_i} \le 10000$.

Sample

Input	Output
2	Case #1: 2

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

Here is an example:



In the sample case, there are two balloons in the sky, and you have 1 energy point to use. The best solution is to immediately spend 1 energy point to move the balloon at position 3, height 3 down to height 2. Once you've done that, it will take 2 time units for both balloons to reach the tower.

Round D APAC Test 2016

A. Dynamic Grid

B. gBalloon

C. IP Address Summarization

D. Virtual Rabbit

Questions asked 1

Submissions

Dynamic Grid

6pt | Not attempted **1392/1881 users** correct (74%)

8pt | Not attempted **1288/1368 users** correct (94%)

gBalloon

9pt | Not attempted **353/666 users** correct (53%)

17pt | Not attempted **266/338 users** correct (79%)

IP Address Summarization

10pt | Not attempted **123/236 users** correct (52%)

19pt | Not attempted **73/118 users** correct (62%)

Virtual Rabbit

11pt | Not attempted **18/166 users** correct (11%)

20pt | Not attempted **3/8 users** correct (38%)

Top Scores

nhho	100
sundar95	80
Shaon	80
ajkrish95	80
ojas.deshpande	80
NAFIS	69
JunoYu	69
wcwswsws	69
karanaggarwal	69
VotBear	69

Problem C. IP Address Summarization

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

Small input 10 points	Solve C-small
Large input 19 points	Solve C-large

Problem

An IP (Internet Protocol) address is a number that is assigned to each device on the Internet. At the time being, most devices use version four of this protocol (IPv4). An IPv4 address is a 32-bit string. IPv4 addresses are normally represented in dot-decimal notation, which consists of four decimal numbers called octets, each ranging from 0 to 255 (inclusive), separated by dots, e.g., 172.16.254.1. Each octet represents a group of 8 bits (one byte) of the address. The first 8 bits of the string (when interpreted as an unsigned integer, with the most significant bit first) form the first octet, the next 8 bits form the second octet, and so on.

An IP subnet addresses is used to represent a group of devices that belong to the same network. IP subnet addresses are expressed in the format of an IP address, followed by a slash and then a prefix length ranging from 0 to 32. A subnet address stands for all IP addresses that have the same first P bits of the given address, where P is the prefix length. For example 10.8.0.0/9 represents 2^{23} addresses that all have `000010100` (the first nine bits of 10.8.0.0) as their first 9 bits, that is, 10.0.0.0 through 10.127.255.255. Note that 10.8.0.0/9 and, for example, 10.0.0.0/9 (or any other address within the subnet) would be equivalent ways to refer to the same subnet, because those addresses start with the same nine bits.

A subnet is *normalized* if the bits of the address other than the prefix are all zeroes. For example, 10.8.1.0/24 and 10.8.1.2/24 represent the same subnet, but 10.8.1.0/24 is normalized. The normalization of 255.255.255.255/13 is 255.248.0.0/13.

You will be given a list of subnet addresses, and you must output the shortest ordered list of subnets such that all the addresses are normalized and an address belongs to some subnet in the input if and only if it belongs to some subnet in the output.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each begins with one line with an integer **N**, the number of subnets, and is followed by **N** more lines, each of which has a subnet addresses. Each subnet address is of the form `A.B.C.D/P`, where A, B, C, and D are integers from 0 to 255, inclusive, and P is an integer from 0 to 32, inclusive. No integer (apart from 0) has leading zeroes.

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). Then output a list of subnet addresses, one per line, meeting the conditions described above. These addresses must be normalized and must be ordered. An address X comes before another address Y if X's first integer is smaller than Y's first integer, or if X and Y have the same first integer but X's second integer is smaller than Y's second integer, and so on.

Note that the requirements of the problem guarantee that there is a single unique answer for each test case.

Limits

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

Small dataset

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

Large dataset

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

Sample

Input	Output
3	Case #1:

```
2                       10.0.0.0/8
10.1.2.3/8              Case #2:
10.2.3.4/17             10.0.0.0/8
2                       Case #3:
10.2.3.4/9              52.22.138.0/23
10.128.2.3/9            52.56.134.128/26
10                      52.227.80.0/22
224.147.224.186/18      58.32.0.0/12
58.45.85.53/14          83.248.0.0/13
52.56.134.139/26        224.147.192.0/18
52.227.82.227/22        238.223.58.128/27
83.250.251.44/13
83.250.12.64/16
58.40.52.11/14
52.22.138.56/23
238.223.58.151/27
58.32.52.11/13
```

Problem D. Virtual Rabbit

Small input 11 points	Solve D-small
Large input 20 points	Solve D-large

Problem

Alice just bought a virtual pet rabbit. The rabbit hops around on a screen and can be "fed" by pressing a button. Alice is fond of the rabbit, but she is also busy, and doesn't want to spend too much time taking care of it. However, if the rabbit goes without "food" for too long, it "dies" and Alice loses the game.

Every day, Alice gets up at time **G**, goes to work at time **W**, returns home at time **H**, and goes to bed at time **B**. Alice cannot feed the rabbit while she is at work or asleep -- that is, in the intervals [**W**, **H**) and [**B**, **G**). Note that times **W** and **B** themselves are not valid feeding times, whereas times **H** and **G** are. In any other second, Alice can either push a button to instantly feed the rabbit, or not push the button. Between every two seconds, the rabbit determines the number of consecutive seconds in which it has not been fed, and "dies" if that duration is equal to **X**.

It is currently 00:00:00 on Day 0, and the rabbit has just been delivered to Alice's house by the mail service. (The mail carrier pushes the button at 00:00:00, even if Alice is asleep, and then leaves.) Alice wants to make sure the rabbit is still "alive" at 00:00:00 on day **D**. What is the minimal number of times that she needs to feed the rabbit, if she can keep the rabbit "alive" at all?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of 6 lines. The first 5 lines represent the times **G**, **W**, **H**, **B**, and **X** in "hh:mm:ss" format. The last line consists of one integer **D**.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the minimal number of times Alice needs to feed the rabbit. If it's impossible for the rabbit to be alive at 00:00:00 on day **D**, output -1.

Limits

$1 \le \mathbf{T} \le 100$.
It's guaranteed that Alice always goes to bed before midnight and gets up at or after midnight. **G**, **W**, **H**, and **B** are in strictly increasing order within the same day.
$00{:}00{:}00 \le \mathbf{G} < \mathbf{W} < \mathbf{H} < \mathbf{B} \le 23{:}59{:}59$.
$00{:}00{:}00 < \mathbf{X} \le 23{:}59{:}59$.

Small dataset

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

Large dataset

$1 \le \mathbf{D} \le 10^{14}$.

Sample

Input	Output
3 08:00:00 09:00:00 18:00:00 22:00:00 12:00:00 100 08:00:00 09:00:00 18:00:00 22:00:00 01:00:00 1 00:00:00 12:00:00	Case #1: 200 Case #2: -1 Case #3: 86401

```
12:00:01
23:59:59
00:00:02
2
```

In sample case #1, Alice could feed the rabbit at 08:00:00 and 20:00:00 every day.

In sample case #2, the poor rabbit will be "dead" before Alice even wakes up on Day 0.

Problem A. Lazy Spelling Bee

Small input 5 points	Solve A-small
Large input 8 points	Solve A-large

Problem

In the Lazy Spelling Bee, a contestant is given a target word W to spell. The contestant's answer word A is acceptable if it is the same length as the target word, and the i-th letter of A is either the i-th, (i-1)th, or (i+1)th letter of W, for all i in the range of the length of A. (The first letter of A must match either the first or second letter of W, since the 0th letter of W doesn't exist. Similarly, the last letter of A must match either the last or next-to-last letter of W.) Note that the target word itself is always an acceptable answer word.

You are preparing a Lazy Spelling Bee, and you have been asked to determine, for each target word, how many distinct acceptable answer words there are. Since this number may be very large, please output it modulo 1000000007 ($10^9 + 7$).

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of one line with a string consisting only of lowercase English letters (a through z).

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the number of distinct acceptable answer words, modulo $10^9 + 7$.

Limits

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

Small dataset

1 ≤ length of each string ≤ 5.

Large dataset

1 ≤ length of each string ≤ 1000.

Sample

Input	Output
4 ag aa abcde x	Case #1: 4 Case #2: 1 Case #3: 108 Case #4: 1

In sample case #1, the acceptable answer words are aa, ag, ga, and gg.

In sample case #2, the only acceptable answer word is aa.


Powered by
Google Cloud Platform

Problem B. Robot Rock Band

Small input 6 points	Solve B-small
Large input 14 points	Solve B-large

Problem

You're the manager of Xorbitant, the world's first robot rock band. There will be four positions in the band, and there are **N** robots auditioning for each position. (No robot is auditioning for more than one position.) Every robot has a number, and multiple robots might have the same number, just as two people can have the same name.

You know from market research that your robot audiences won't care how well the robot band members make music, how handsome they are, or what scandalous things the tabloids say about them. Instead, the audience will be checking to see whether the four members' numbers, when bitwise XORed together, equal a certain trendy number **K**.

How many different sets of four robots (one for each position) is it possible to choose so that the band will have this property? More specifically, given four lists A, B, C, D containing **N** numbers each, how many ways are there to choose one number *a* from list A, one number *b* from list B, and so on, such that *a*^*b*^*c*^*d* = **K**? (Here ^ represents the bitwise XOR operation.)

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each case begins with one line with two space-separated integers, **N** and **K**, as described above. Then, four more lines follow. Each has **N** space-separated integers and represents the ID numbers of the robots auditioning for a certain position in the band.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the number of different bands that meet the conditions.

Limits

$1 \le \mathbf{T} \le 10$.
$0 \le \mathbf{K} \le 10^9$.
$0 \le$ all robot numbers $\le 10^9$.

Small dataset

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

Large dataset

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

Sample

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

In sample case #1, in order to get a combined bitwise XOR of 3, the robot chosen from the second list must be 2, and the robot chosen from the fourth list must be 1. For the first and third lists, either of the two 0 robots can be chosen, so there are 2 * 2 = 4 possible bands that meet the criteria. Note that

even though all of these bands are of the form (0, 2, 0, 1), they are considered different because the selections from the lists were different.

Round E APAC Test 2016

A. Lazy Spelling Bee
B. Robot Rock Band
C. Not So Random
D. Sums of Sums

Questions asked

Problem C. Not So Random

Problem

There is a certain "random number generator" (RNG) which takes one nonnegative integer as input and generates another nonnegative integer as output. But you know that the RNG is really not very random at all! It uses a fixed number **K**, and always performs one of the following three operations:

- with probability **A**/100: return the bitwise AND of the input and **K**
- with probability **B**/100: return the bitwise OR of the input and **K**
- with probability **C**/100: return the bitwise XOR of the input and **K**

(You may assume that the RNG *is* truly random in the way that it chooses the operation each time, based on the values of **A**, **B**, and **C**.)

You have **N** copies of this RNG, and you have arranged them in series such that output from one machine will be the input for the next machine in the series. If you provide **X** as an input to the first machine, what will be the expected value of the output of the final machine in the series?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of one line with six integers **N**, **X**, **K**, **A**, **B**, and **C**. Respectively, these denote the number of machines, the initial input, the fixed number with which all the bitwise operations will be performed (on every machine), and 100 times the probabilities of the bitwise AND, OR, and XOR operations.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the expected value of the final output. y will be considered correct if it is within an absolute or relative error of 10^{-9} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

$1 \le \mathbf{T} \le 50$.
$0 \le \mathbf{A} \le 100$.
$0 \le \mathbf{B} \le 100$.
$0 \le \mathbf{C} \le 100$.
$\mathbf{A}+\mathbf{B}+\mathbf{C} = 100$.

Small dataset

$1 \le \mathbf{N} \le 10$.
$0 \le \mathbf{X} \le 10^4$.
$0 \le \mathbf{K} \le 10^4$.

Large dataset

$1 \le \mathbf{N} \le 10^5$.
$0 \le \mathbf{X} \le 10^9$.
$0 \le \mathbf{K} \le 10^9$.

Sample

Input	Output
3 1 5 5 10 50 40 2 5 5 10 50 40 10 15 21 70 20 10	Case #1: 3.0000000000 Case #2: 3.6000000000 Case #3: 15.6850579098

In sample test case #1, the final output will be 5 if AND or OR happens and 0 if XOR happens. So the probability of getting 5 is (0.1 + 0.5) and the probability

of getting 0 is 0.4. So the expected final output is 5 * 0.6 + 0 * 0.4 = 3.

In sample test case #2, the final output will be 5 with probability 0.72, and 0 otherwise.

Problem D. Sums of Sums

Problem

Alice presented her friend Bob with an array of **N** positive integers, indexed from 1 to **N**. She challenged Bob with many queries of the form "what is the sum of the numbers between these two indexes?" But Bob was able to solve the problem too easily.

Alice took her array and found all **N***(**N**+1)/2 non-empty subarrays of it. She found the sum of each subarray, and then sorted those values (in nondecreasing order) to create a new array, indexed from 1 to **N***(**N**+1)/2. For example, for an initial array [2, 3, 2], Alice would generate the subarrays [2], [3], [2], [2, 3], [3, 2], and [2, 3, 2] (note that [2, 2], for example, is **NOT** a subarray). Then she'd take the sums -- 2, 3, 2, 5, 5, 7 -- and sort them to get a new array of [2, 2, 3, 5, 5, 7].

Alice has given the initial array to Bob, along with **Q** queries of the form "what is the sum of the numbers from index $\mathbf{L_i}$ to $\mathbf{R_i}$, inclusive, in the new array?" Now Bob's in trouble! Can you help him out?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case begins with one line with two space-separated integers **N** and **Q**, denoting the number of elements in the initial array and the number of Alice's queries. Then, there is one line with **N** space-separated integers, denoting the elements of Alice's initial array. Finally, there are **Q** more lines with two space-separated integers each: $\mathbf{L_i}$ and $\mathbf{R_i}$, the inclusive index bounds for the i-th query.

Output

For each test case, output one line with `Case #x:`, where **x** is the test case number (starting from 1). Then output **Q** more lines, each with one integer, representing the answers to the queries (in the order they were asked).

Limits

$1 \le \mathbf{T} \le 10$.
$1 \le \mathbf{Q} \le 20$.
$1 \le$ each element of the initial array ≤ 100.
$1 \le \mathbf{L_i} \le \mathbf{R_i} \le N*(N+1)/2$.

Small dataset

$1 \le \mathbf{N} \le 10^3$.

Large dataset

$1 \le \mathbf{N} \le 200000$.

Sample

Input	Output
1 5 5 5 4 3 2 1 1 1 1 10 1 15 3 8 4 11	Case #1: 1 45 105 26 48

In sample case #1, Alice's new array would be: [1, 2, 3, 3, 4, 5, 5, 6, 7, 9, 9, 10, 12, 14, 15].