University of Computer Studies, Yangon

2018 ICPC Asia-Yangon Regional Programming Contest

December 8-9, 2018

regionals

icpc International Collegiate Programming Contest

icpc.foundation

Welcome to the 2018 Regional Programming Contest. Before you start the contest, please be aware of the following notes:

The Contest

1. There are twelve (12) problems in the packet, using letters A-L. These problems are NOT necessarily sorted by difficulty. As a team's solution is judged correct, the team will be awarded a balloon. The balloon colors are as follows:

Problem	Problem Name	Balloon Color
A	**Forloop Hazard**	Purple
B	**Rabbit and Carrot**	Green
C	**Food Exchange**	Blue
D	**AND Queries**	Pink
E	**Telecom Towers**	Brown
F	**The Bit Game**	Red
G	**Spreading Message**	Yellow
H	**The Block Chain**	Orange
I	**Consistency Checker**	Black
J	**Least Common Multiple**	Cyan
K	**Mountaineering**	White
L	**Molecular Arrangement**	Gold

2. Solutions for problems submitted for judging are called runs. Each run will be judged. The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
Yes	Your submission has been judged correct.
Wrong	Answer Your submission generated output that is not correct.
Output Format Error	Your submission's output is not in the correct format or is misspelled.
Incomplete Output	Your submission did not produce all of the required output.
Excessive Output	Your submission generated output in addition to or instead of what is required.
Compilation Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Time-Limit Exceeded	Your submission did not solve the judges' test data within 60 seconds.
Other-Contact Staff	Contact your local site judge for clarification.

3. A team's score is based on the number of problems they solve and penalty points, which reflect the amount of time and number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charged equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved. Teams are ranked first by the number of problems solved and then by the fewest penalty points.

4. This problem set contains sample input and output for each problem. However, the judges will test your submission against longer and more complex datasets, which will not be revealed until after the contest. Your major challenge is designing other input sets for yourself so that you may fully test your program before submitting your run. Should you receive a judgment stating that your submission was incorrect, you should consider what other datasets you could design to further evaluate your program.

5. In the event that you feel a problem statement is ambiguous or incorrect, you may request a clarification. Read the problem carefully before requesting a clarification.
   If a clarification is issued during the contest, it will be broadcast to all teams.
   If the judges believe that the problem statement is sufficiently clear, you will receive the response, "No response, read problem statement."
   If you receive this response, you should read the problem description more carefully. If you still feel there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found.
   You may not submit clarification requests asking for the correct output for inputs that you provide, e.g., "What would the correct output be for the input ...?" Determining that is your job unless the problem description is truly ambiguous. Sample inputs may be useful in explaining the nature of a perceived ambiguity, e.g., "There is no statement about the desired order of outputs. Given the input: . . . , would both this: . . . and this: . . . be valid outputs?".

6. Runs for each particular problem will be judged in the order they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for problem B followed by a run for problem C, but receive the response for C first.
   Do not request clarifications on when a response will be returned. If you have not received a response for a run within 30 minutes of submitting it, **you may have a runner ask the local site judge to determine the cause of the delay**. Under no circumstances should you ever

submit a clarification request about a submission for which you have not received a judgment. If, due to unforeseen circumstances, judging for one or more problems begins to lag more than 30 minutes behind submissions, a clarification announcement will be issued to all teams. This announcement will include a change to the 30 minute time period that teams are expected to wait before consulting the site judge.

7. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification.

8. The submission of code deliberately designed to delay, crash, or otherwise negatively affect the contest itself will be considered grounds for immediate disqualification.

Your Programs

9. All solutions must read from standard input and write to standard output. In C this is ***scanf/printf***, in C++ this is ***cin/cout***, and in Java this is ***System.in/System.out***. The judges will ignore all output sent to standard error (***cerr*** in C++ or ***System.err*** in Java). You may wish to use standard error to output debugging information. From your workstation you may test your program with an input file by redirecting input from a file:

```
program < file.in
```

10. Unless otherwise specified, all lines of program output
    - must be left justified, with no leading blank spaces prior to the first non-blank character on that line,
    - must end with the appropriate line terminator (\n, endl, or println()), and
    - must not contain any blank characters at the end of the line, between the final specified output and the line terminator.

    You must not print extra lines of output, even if empty, that are not specifically required by the problem statement.

11. Unless otherwise specified, all numbers in your output should begin with the minus sign (-) if negative, followed immediately by 1 or more decimal digits. If the number being printed is a floating point number, then the decimal point should appear, followed by the appropriate number of decimal digits. For output of real numbers, the number of digits after the decimal point will be specified in the problem description (as the "precision").
    All floating point numbers printed to a given precision should be rounded to the nearest value. For example, if 2 decimal digits of precision is requested, then 0.0152 would be printed as "0.02" but 0.0149 would be printed as "0.01".
    In simpler terms, neither scientific notation nor commas will be used for numbers, and you should ensure that you use a printing technique that rounds to the appropriate precision.

12. All input sets used by the judges will follow the input format specification found in the problem description. You do not need to test for input that violates the input format specified in the problem.

13. All lines of program input will end with the appropriate line terminator (e.g., a linefeed on Unix/Linux systems, a carriage return-linefeed pair on Windows systems).

14. If a problem specifies that an input is a floating point number, the input will be presented according to the rules stipulated above for output of real numbers, except that decimal points

and the following digits may be omitted for numbers with no non-zero decimal portion. Scientific notation will not be used in input sets unless a problem explicitly allows it.

15. Every effort has been made to ensure that the compilers and run-time environments used by the judges are as similar as possible to those that you will use in developing your code. With that said, some differences may exist. It is, in general, your responsibility to write your code in a portable manner compliant with the rules and standards of the programming language. You should not rely upon undocumented and non-standard behaviors.
   (a) One place where differences are likely to arise is in the size of the various numeric types. Many problems will specify minimum and maximum values for numeric inputs and outputs. You should write your code with the understanding that, on the judges' machines:
      - A C++ int, a C++ long, and a Java int are all 32-bits wide.
      - A C++ long long and a Java long are 64-bits wide.
      - A float in both languages is a 32-bit value capable of holding 6-7 decimal digits, though many library functions will be less accurate.
      - A double in both languages is a 64-bit value capable of holding 15-16 decimal digits, though many library functions will be less accurate.

      The data types on your own machines may differ in size from these, but if you follow the guidelines above in choosing the types to hold your numbers, you can be assured that they will suffice to hold those values on the judges' machines.

   (b) Another common source of non-portability is in C/C++ library structures. Although the C & C++ standards are very explicit about which header file must declare certain std symbols, the standards do not prohibit other headers from duplicating or loading extra symbols.
   For example, if your program uses both ***cout*** and ***ifstream***, you might find that your code compiles if you only ***#include <fstream>***, because, as it happens, on your machine the ***fstream*** header ***#includes*** the ***iostream*** header where ***cout*** is properly declared. However, you cannot rely upon the judges' machines having libraries with the same structure. So it is your responsibility to ***#include*** the appropriate headers for whatever std library features you use.

Good luck, and HAVE FUN!!!

Problem A: Forloop Hazard

Run Time Limit: 3 sec

Look at the following C Program. What will it print for an input **high**? Of course one can run the program and find out but if the value of variable **high** is quite large then you have to wait for centuries to get the output. So you will have to find a clever way to find the output.

```
#include<stdio.h>
#include<math.h>
#include<stdint.h>
int64_t MODV = 10000019;
int main(void)
{
    int64_t t, m, i, j, ans, high;
    double dans;
    while(1)
    {
          scanf("%lld",&high);
          if(high==0) break;
          for(m=1,ans=0;m<=high;m++)
          {
             for(i=1;i<=m;i++)
               {
                   for(j=1;j*j*j<=i;j++)
                 ans=(ans+1) % MODV;
               }
          }
          printf("%lld\n",ans);
    }
}
```

C Program

Input
The input file contains at most **10000** lines of input. Each line contains an integer denoting a possible value of input variable **high** ($0 < high \le 10^{17}$). Input is terminated by a line containing a single zero. This line should not be processed.

Output
For each line of input produce one line of output. This line contains the output that the program above will produce. As the value of output can be quite large so you are requested to print the modulo **10000019** value of it.

Sample Input	Sample Output
10	61
1000	2974675
10000	7734691
0	

Problem B: Rabbit and Carrot

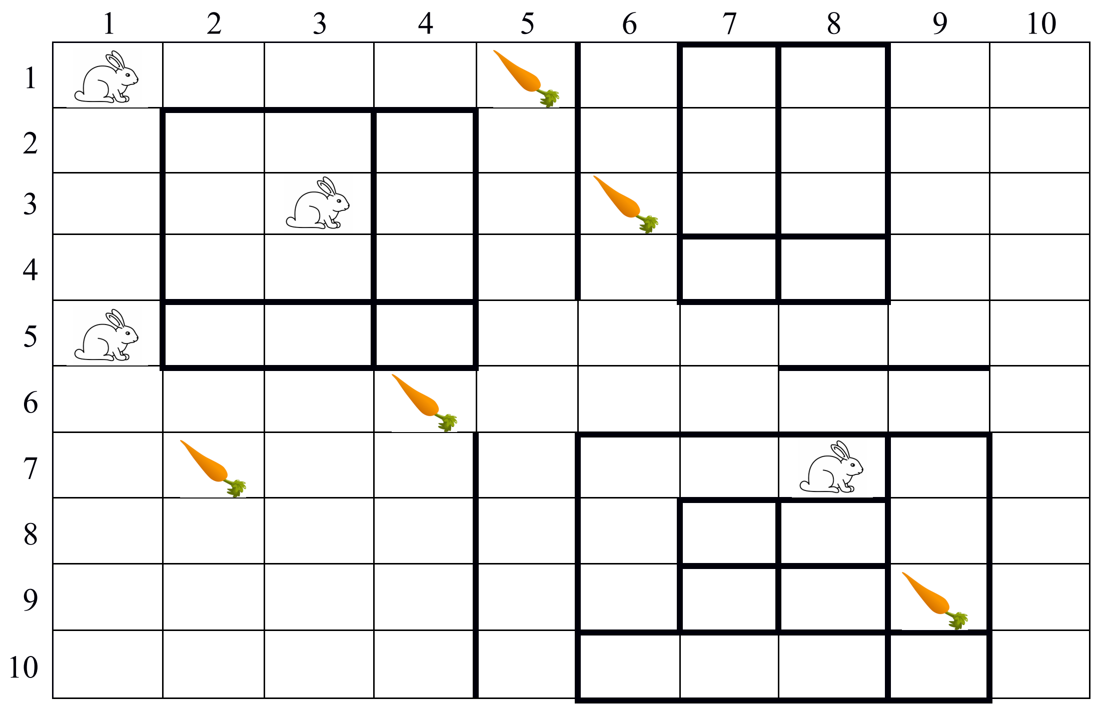
Run Time Limit: 3 sec

A field can be represented as rectangular grid consisting of rows and columns. The rows of the field are numbered from top to bottom and the columns are numbered from left to right. There will be *n* rabbits in the field and each rabbit occupying a unit square represented by row and column number (*r*, *c*).The field also contains *m* carrots and each carrot occupying a unit square. Finally, the field contains *k* rectangular blocks.

Assume that no two blocks intersect or touch. However, a block may contain other block inside the enclosed area. Rabbits can only move in two directions – down or right. The rabbit cannot pass through the block but can go through the squares occupied by others rabbits or carrots.

Your job is to write the program, to find the total numbers of carrots for each rabbits, reachable from its current position.

The following figure shows the sample simulation of rabbits and carrots.



Input

Input contains three portions – the first is the rectangular blocks, the second portion represents for the carrots and the third portion for rabbits.

For the first portion of input – the first line contains an integer *k* (0≤ *k* ≤200) that represents the number of rectangular blocks. Each of the following *k* lines contains four integers r1, c1, r2, c2 (1≤r1, c1, r2, c2 ≤1000) for representing a block. The input r1 and c1 are the coordinates (row and column) of the upper-left corner square, while r2 and c2 are the coordinates of the lower-right corner. No two blocks will intersect or touch.

For the second portion of input – the first line contains an integer *m* (0≤*m*≤200) that represents the number of carrots. Each of the following *m* lines contains two integers *r* and *c* (1 ≤ *r*, *c* ≤ 1000) – the location of for each carrots. No two carrots will occupy the same location.

For the third portion of input – the first line contains an integer *n* ($1 \le n \le 200$) that represents the number of rabbits. Each of the following *n* lines contains two integers *r* and *c* ($1 \le r, c \le 1000$) – the location of for each rabbits. No two rabbits will occupy the same location, and no carrot and rabbit will occupy the same location.

Output

Output should consist of n lines. Each line should contain a single integer for the total number of carrots for each rabbit reachable from its location.

Sample Inputs	Sample Outputs

4
2 2 5 4
1 7 4 8
7 6 10 9
8 7 9 8
5
7 2
6 4
1 5
3 6
9 9
4
1 1
5 1
3 3
7 8
	4
	2
	0
	1

Problem C: Food Exchange

Run Time Limit: 3 sec

In an ancient world, food and cuisine are among the first aspects of identity to meet clash and enrich each other countries. Therefore each country has a food exchange center. But sometimes, some of the countries don't want to engage in a process of exchange because they don't like the other's cultures. A country is called **an articular country** if they stop the food exchange or disconnect with others and closed the gate, some of the countries cannot also exchange or connect to the rest ones. There may be the countries, which cannot exchange any food with all others and they must be stand-alone unfortunately un-survival. In the following graphs, there are 8 countries in food exchange and 9 paths they can connect each other. If any one of the country number 2, 3, 4 and 5 stops the exchange and disconnects with others, they will be articular countries. The country number 6 and 7 will then be stand-alone countries if all the articular countries stop exchange.

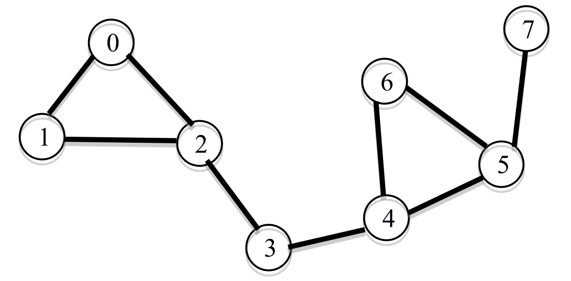


Figure 1: All the countries exchange food according to the connected path

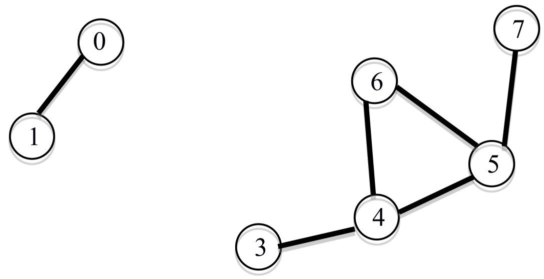


Figure 2: Country No. 2 stop exchange disconnect, 0 and 1 cannot get any food from the rest of the others except each other.

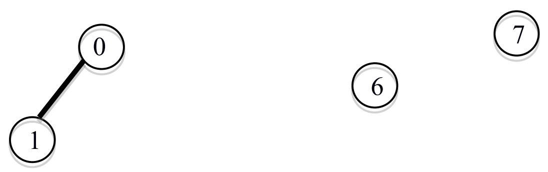


Figure 3: Articular Country No. 2, 3, 4 and 5 stop exchange or disconnect, 6 and 7 become stand-alone country

Input

- First line contains **T**, the number of test cases.
- The second line shows the two separated integers such as number of countries in food exchange, **N** and the number of paths they connect each other **P**.
- This is followed by the two separated integers in **v** and **w** such as the path between the two countries.

Output

- Print the case number and the stand-alone (Un-Survival) countries
- If there is no stand-alone country, just print the case number.

Constraint

- $1 \leq T \leq 10^5$
- $1 \leq N \leq 10^5$
- $1 \leq P \leq 10^5$
- $0 \leq v, w \leq N$

Sample Inputs	Sample Outputs

2
8 9
0 1
0 2
1 2
2 3
3 4
4 5
4 6
5 6
5 7
5 6
0 1
0 2
0 3
1 4
2 4
3 4
	Case 1: 6 7
	Case 2:

Problem D: AND Queries

Run Time Limit: 10 sec

Given an array A of N integers, you are required to solve Q queries. Each query consists of a positive integer V and a non-negative integer K. For each query, find out how many numbers in the array A have exactly K common one bits with the number V.

In other words, for each query, you need to calculate how many numbers are there in the array such that A_i & V have exactly K bits set in its binary representation, where & denotes the bitwise AND operation.

Note that since the input can be huge, they will be generated randomly using a pseudorandom generator, whose parameters will be given as input. Also for similar reasons, it is not required to output the result for every query, rather compute the sum of this value for all queries and output this sum. More specifically, for the i-th query, let C_i be the count of integers in A having exactly K common one bits with V_i. Then it is required to output the sum of all C_i only.

Input

The first line contains an integer T, denoting the number of test cases. Each test case contains six space separated integers in the order: seed, N, Q, mod_A, mod_V, mod_K. Afterwards, the input for each test case will be generated as described by the python code below.

```
        def random():
            seed = (seed * 997 + 29) % 2117566807  # Watch out for overflow if
    in C++
            return seed;

        for i in range(N):
            A[i] = random() % mod_A

        for i in range(Q):
            V = random() % mod_V
            K = random() % mod_K
```

Note that the seed is a global variable which gets updated after each random call, with the initial value being given as input.

Output

For each test case, output the case number followed by the required output. Please refer to the sample input/output section for more clarity of the format.

Constraints

$1 \leq T \leq 25$

$1 \leq N, Q \leq 250{,}000$

$0 \leq seed < 2117566807$

$1 \leq mod\_A, mod\_V \leq 250{,}000$

$1 \leq mod\_K \leq 19$

Sample Inputs	Sample Outputs

2
1 10 10 4 4 3
0 100 1000 10000 100000 10
	Case 1: 26
	Case 2: 10260

Explanation:

For the first case, **A** = [2, 3, 0, 1, 1, 2, 2, 3, 1, 0], and the queries are:

V_i	K_i	C_i
2	0	5
0	2	0
3	1	6
1	1	5
0	1	0
0	2	0
2	2	0
0	0	10
0	2	0
1	2	0

Problem E: Telecom Towers

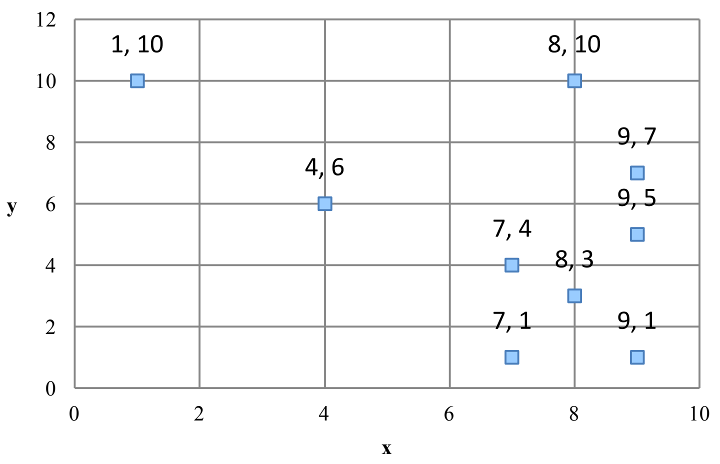
Run Time Limit: 3 sec

Telecom Myanmar is a telecommunications infrastructure provider to build and manage telecom towers. Planning manager of Telecom Myanmar first makes a survey for main places in a region where mobile signal have to reach. It is not necessary to build a tower in each place. It only needs mobile signal to cover all places. Therefore, towers in some places can be reduced to be built. Planning manager needs to decide where and how many towers should be built to achieve minimum cost.

Therefore, your work is to determine minimum number of towers that needs to be built so that these towers will cover the main places of the region.

The working range of each tower (the range which mobile devices connect reliably to the tower) is *s miles* away from it.

For example, the following figure shows the coordinates of main places that mobile signal have to reach.



If the working range of a tower is *4 miles*, he has to build at least three towers (at the locations, (7, 4), (1, 10) and (8, 10) as a sample) to cover all places.

If the working range of a tower is *6 miles*, two towers are required to be built (at the locations, (9, 1) and (4, 6) as a sample) that signal can reach to all places.

Input

First line consists of an integer, n, ($1 \leq n \leq 1000$), number of test cases.

Each test case contains several lines of coordinates of places, each containing two integers, x and y, ($0 \leq x, y \leq 1000$). This will be terminated by “-1 -1”.

Last line of each test case contains an integer, s, specifying signal coverage of a tower (in miles).

Output

For each test case, display minimum number of towers required to be built that mobile signal will reach to all input places.

Sample Inputs	Sample Outputs

2
7 4
9 5
9 1
8 3
9 7
7 1
4 6
1 10
8 10
-1 -1
4
8 6
5 4
3 5
3 8
3 3
10 6
-1 -1
3
	3
	2

Problem F: The Bit Game

Run Time Limit: 1 sec

Two players, Ko Ko and Bo Bo, are given an integer N to play a game.
The rules of the game are as follows:

1. In one turn, a player can **swap any 2 bits** of N in its binary representation to make a new N.
2. In one turn, a player has to make a number **strictly less than N**.
3. Ko Ko always takes first turn.
4. If a player cannot make a move, he loses.

Assume that both the players play optimally.

Input

First line of input contains a single integer T denoting the number of test cases. The only line of each test case contains an integer **N**.

Constraints :

```
1 <= T <= 200
1 <= N <= 10^12
```

Output

For each test case, print "1" if Ko Ko wins, else print "2" in a new line (without quotes "").

Sample Inputs	Sample Outputs
3 8 1 42	1 2 2

Explanation:

Case 1 :

N = 8
N = 1000 (binary)
Ko Ko swaps the 1st and 4th bit.
100**0**
N = 0001
Bo Bo cannot make a move, so Ko Ko wins.

Case 2 :

N = 1
Ko Ko cannot make a move, so Bo Bo wins.

Case 3 :

N = 42
N = 101010 (binary)
Ko Ko swaps the 1st and 6th bit.
10101**0**
N = 1011
Bo Bo swaps 1st and 2nd bit.
1011
N = 111
Ko Ko cannot make a move, so Bo Bo wins.

Problem G: Spreading Message

Run Time Limit: 1 sec

Sai is a second year student in University of Computer Studies, Yangon. He enjoys in developing mobile applications. He has recently developed a sms-like app. He installed it on mobile devices of his friends and tested. His apps works as: once a message is sent using his app, then the app automatically sends it to all contacts marked as favorite in the mobile device. It takes ***t1*** seconds to arrive to each of these contacts' devices. Once these devices receive the message, the app continues sending the message to their favorite contacts. This process continues until there is no favorite contact.

Sai wants to know that who will receive the message after ***t2*** seconds it is sent from one of a group of people who installed the app on their mobile devices.

Input

The first line of the input contains **an integer**, C***(1 ≤ C ≤ 100)*** that represents number of test cases.

In each test case, first line contains **two integers**, ***n (1 ≤ n ≤ 1 000)*** specifying the number of people and ***t1*** (***1 ≤ t1 ≤ 100)*** representing the time taken (in seconds) to receive a message from a person to his/her favorite contacts.

This is followed by ***n lines***. Each contains ***a string of words (names)*** separated by a single space. Names starting from second word are favorite contacts' names of the first one.

(For example, consider the following input:

Steve Mary Rose Ted

This implies that favorite contacts of Steve are Mary, Rose and Ted.)

Finally a number of queries follow until it finds **'end'**. Each query contains ***a word (name)*** representing a person who starts sending a message and ***an integer,t2 (1 ≤ t2 ≤ 100000),*** representing the time interval in seconds after the message is sent.

Output

Display all people who receive the message after *t2*seconds it is sent in ascending order for each query. If no person receives the message, then print "NO PERSON". A blank line follows at the end of each test case's output.

Sample Inputs	Sample Outputs
2 7 2 John Mary Wendy Mary Rose Steve Rose Steve Ted Mary Steve Ted Wendy Rose Ted Rose Wendy Wendy Steve Ted Smith John 5 Smith 4 Rose 3 Ted 4	Mary Rose Steve Ted Wendy NO PERSON Mary Steve Ted Mary Rose Steve Wendy NO PERSON Rose Ted Wendy a d f b c d NO PERSON a b d e f g h i j a b c d e f g h i a b c d f i j

Wendy 1 Steve 2 end 10 3 a b c d b a g c a h d a e f e d f d j g b h c i i h j j f i e 6 a 3 b 2 c 12 j 15 h 10 end	

Problem H: The Block Chain

Run Time Limit: 3 sec

There are many mobile devices which are connected each other around the world. Crypto currency system is rapidly developed and the security concern becomes the important challenge. The block chain is derived from block chan. It is a growing list of records call blocks, which are linked using cryptography. Each block contains a cryptographic **hash** of the previous block and transaction **data**. The hash is the integrity check function, to verify the message is the original. The hash value is depended on the original data but it generates the fixed size value.

To secure the communication for the data transaction, implement the block chain system. The system uses MD5 hash function for generation the hash code for each block. Each block must contain **previous hash, new hash and data**. However, it only needs to generate the output for hash code. The MD5 hash function will generate fixed size, 32 characters output.
(Hint: MessageDigest.getInstance("MD5") function can be used from java.security package) .
The sample inputs and outputs are shown in Table 1.

Input

The first line contains the number of test cases. The next line consists of the number of lines **N** for each test case. The next **N** lines are the input text for each test case.

Output

Print each test case number first and then, in the next lines, display the values of Hash, Data and Previous hash in each line. Use curly bracket, { }, to enclose each test case.

Sample Inputs

```
2
3
Hello, it’s first transaction.
Do you protect your data?
Security is very important.
2
Welcome from UCSY.
It is located in Yangon.
```

Sample Outputs

```
Case 1:
{Hash: a500af614393031276366ff4b6ad12ad
Data: Hello, it’s first transaction.
Previous Hash: 0}
{Hash: c241fa6712b3ebd5105da962fd978d82
Data: Do you protect your data?
Previous Hash: a500af614393031276366ff4b6ad12ad}
```

```
{Hash: 665d673b027dc19e37cd146995ea3102
Data: Security is very important.
Previous Hash: c241fa6712b3ebd5105da962fd978d82}
Case 2:
{Hash: 7e3ce1d695ae75cd8b4caf5dbfb94c3f
Data: Welcome from UCSY.
Previous Hash: 0}
{Hash: dded67e9089478b3afd0305183c210be
Data: It is located in Yangon.
Previous Hash: 7e3ce1d695ae75cd8b4caf5dbfb94c3f}
```

Problem I: Consistency Checker

Run Time Limit: 15 sec

You will be given **N** linear inequalities of the following forms:

- ax + by <= c
- ax + by >= c

Here, **a, b, c** are constants and **x, y** are variables.
You have to process **Q** queries. Each query consists of a pair of integers **(x, y)** and you have to check the feasibility of the given system for them.
The query points will be generated using the following code, where Q, initial value of seed, c1 and c2 will be given as input. Note that for each query, the seed is a global variable which will get updated after each call to the function get_random:

```
get_random()
{
        seed = (seed * c1 + c2) % 2117566807;
        return seed;
}
generate()
{
   for i from 0 to Q-1
   {
        x[i] = (get_random() - 34492) % 100000;
        y[i] = (get_random() - 34492) % 100000;
   }
}
```

Note that, x % y is x - (Int(x/y))*y. Int(x/y) is the integer part of division x/y.
For example, -7 % 2 will be -7 - (Int(-7/2)) * 2 = -7 - (Int(-3.5)) * 2 = -7 - (-3) * 2 = -7 + 6 = -1

For each test case, generate an array **A** of length **Q**. Where, A[i] is 1 if i'th query satisfies the system, 0 if it doesn't. Output the hash of the array using the following function. Be careful about the data types you choose in your preferred language to avoid overflow.

```
hash_output()
{
        hash = 5731;
        for i from 0 to Q-1
                hash = (hash * 31 + A[i]) % 1000000007;
        return hash;
}
```

Input

First line will contain a single integer **T** (0 < **T** <= 111), denoting the number of test cases.

For each test case, the first line will contain five integers **N (**0 < **N** <= 100), **Q, seed, c1** and **c2.** The next **N** lines will contain two integers **a**, **b** followed by a string and an integer **c**. The string will be either "<=" or ">=".

Constraints:

- Absolute value of a, b, c <= 2*10^5
- At least one of a, b is non zero
- Absolute value of seed <= 10^5
- 0 < c1, c2 <= 1000
- 0 < Q <= 10^6

Output

For each case, output the required hash in a single line. Please check the sample for more clarity.

Sample Inputs	Sample Outputs
1 5 3 34495 1 2 20 5 >= 4 4 3 >= 2 1 5 >= 7 10 9 >= 12 12 5 >= 45	170733214

Explanation:

In the sample, the generated query points are (5, 7), (9, 11) and (13, 15). All of those satisfies the system. So the string would be "111" and the desired hash is 170733214.

Problem J: Least Common Multiple

Run Time Limit: 1 sec

The Least Common Multiple (LCM) of some numbers is the smallest number that the numbers are factors of. Like the LCM of 3 and 4 is 12, because 12 is the smallest number 3 and 4 are both factors for.

- Eg1. the LCM of 6 and 9 is 18, because 18 is the smallest number which is divisible by both 6and 9. i.e. common multiple to both 6 and 9.
- Eg2. the LCM of 3,7 and 9 is 63, because 63 is the smallest number that all of the numbers divide into evenly.
- Eg3. the LCM of 12,2 and 3 is 12, because 12 is the smallest common multiple of 12, 2, 3.

Your task is to output the least common multiple for given three positive numbers.

Input

The first line contains integer T(1<=T<=10) which is the number of test cases. The input for each test case consists of one line that contains the comma-separated three positive numbers, n. Each number, n, should be in range 0<n<1000.

Output

For each test case, print a number that represents the Least Common Multiple of given three numbers. See the samples for the exact format of output.

Sample Inputs	**Sample Outputs**
10	132
4,12,11	105
3,5,7	12
12,2,3	21
3,1,7	12
2,4,6	9700
100,20,97	21780
11,180,121	75570950
91,977,850	8208
38,54,16	63
3,7,9	

Problem K: Mountaineering

Run Time Limit: 3 sec

Mountaineering is a very adventurous sport to attain high points in mountainous regions. On some mountain ranges, the heights could vary very sharply. To be safe, the climber should not choose very height difference with the current location.
Suppose that you want to climb up a mountain range and a height map of this mountain range is provided to you. The height map of a mountain range is described by a 10×10 matrix in which each digit represents the height of the location. You can start from any of the leftmost positions of a given height map. In order to be safe, you will move to an adjacent location by either going up, down, left or right (but not going in a diagonal direction) only if the height difference with the current location is at most 1. The transition from one location to the next is counted.
From a given height map of a mountain range, you have to determine the distance of the shortest viable path between left and right edges.

Input
The first line contains integer T (1<=T<=20) which is the number of test cases. Each test case has a 10×10 matrix which includes 100 digits representing the heights of locations. A line of stars ********** follows each test case for separation.

Output
For each test case, print a single integer that represents the minimum number of transitions to cross the mountain range. If you can't get across, print "impossible".

Sample Inputs	**Sample Outputs**
2	11
8324892342	impossible
1334343293	
3524523454	
2634232043	
0343259235	
3454502352	
4563589024	
7352354256	
9343223234	
2654565443	
**********	
1006314005	
0011012000	
0000001227	
4157978726	
9979999996	
9999999477	
9784422003	
0035496899	
9898468869	
8447987949	
**********	

Problem L: Molecular Arrangement

Run Time Limit: 3 sec

Every substance is composed of one or more molecules. A substance is described as a grid where each molecule is a sequence of cells that starts at one cell and only goes up or right. Examples of molecules are

```
. . a a .
. a a . .
. a . . .
. . . . .
```

```
b b b b b
b . . . .
b . . . .
```

```
. . . .
. . c .
. . . .
. . . .
```

```
. d .
d d .
d . .
. . .
```

A substance has many forms of molecular arrangements where each cell appears in exactly one molecule. Here are all nine possible molecular arrangements of the 2x2 grid, ordered by the number of molecules:

4 molecules: (1 arrangement)

```
a b
c d
```

3 molecules: (4 arrangements)

```
a a     a b     a b     a b
b c     a c     c c     c b
```

2 molecules: (4 arrangements)

```
a a     a b     a a     a b
a b     b b     b b     a b
```

Professor Smith is conducting experiments with molecules. He got one particular molecular arrangement of a substance. After a few days later, he remembers only a few cells (may be zero) of the discovered molecular arrangement and does not remember anymore. His memory is awesome, so he remembers only the cells of the grid one at a time, in row major order. The cells that were remembered are represented by lowercase English letters ('a'-'z'), the unremembered cells are question marks ('?'). Different letters are cells of different molecules, same letters are cells of the same molecule. (As we only use lowercase letters to describe the molecules, the remembered part will contain parts of at most 26 molecules. The original molecular arrangement may have had arbitrarily many molecules.)

Professor Smith needs your help to write a program that counts all molecular arrangements that match the partially remembered grid. Return that count modulo (10^9 + 7).

Notes:

The letters are only used to encode input. A molecule is just a set of squares. Two molecules are distinct if they contain a distinct set of squares. Two molecular arrangements are distinct if they contain a distinct set of molecules.

Input

The first line of the input contains a single integer T, (1<=T<=100) that indicates the number of test cases. Each test case consists of two integers r (1 <= r <= 50) and c (1 <= c <= 50), the number of rows and columns of the grid. This is followed by r lines, each containing c characters that are represented by ('a'-'z') for remembered cells and ('?') for unremembered cells.

Output

For each test case, count all molecular arrangements that match the given grid. Return that count modulo (10^9 + 7)

Sample Inputs	Sample Outputs
6	9
2 2	4
a?	1
??	6
1 3	483683798
???	504
5 4	
aabb	
bbbc	
bdcc	
bdce	
eeee	
2 3	
aaa	
???	
5 7	
???????	
???????	
???????	
???????	
???????	
3 5	
abccd	
eff??	
?????	

Explanation:

Case 1 is explained above.

Case 2:

??? (4 molecular arrangements)

3 molecules:	2 molecules:	1 molecule:
(1 arrangement)	(2 arrangements)	(1 arrangement)
a b c	a a b a b b	a a a

Case 3:

```
aabb
bbbc
bdcc
bdce
eeee
```

(a valid and complete molecular arrangement)

Case 4:

```
aaa
???
```

(6 molecular arrangements)

4 molecules:	3 molecules:	2 molecules:
(1 arrangement)	(3 arrangements)	(2 arrangements)
a a a	a a a a a a a a a	a a a a a a
b c d	a b c b b c b c c	a b b b b b

Case 5:

```
???????
???????
???????
???????
???????
```

Three of the matching molecular arrangements are shown below:

```
bbbbccc        aaaaaaa        abcdefg
bdddddd        bbbbbbb        abcdefg
adeeffg        ccccccc        abcdefg
aeehxxx        ddddddd        abcdefg
aehhzzz        eeeeeee        abcdefg
```

Case 6:

```
abccd
eff??
?????
```

Four of the 504 matching molecular arrangements are shown below:

```
abccd       abccd       abccd       abccd
efffg       effgh       effff       effgg
xyygg       ijklm       ffggg       ggggh
```