ACM Collegiate Programming Contest 2016 (Hong Kong)

CO-ORGANIZERS:

Venue:	Cyberport, Pokfulam
Time:	2016-06-18 [Sat] 1400—1800
Number of Questions:	7

(This is a blank page.)

Problem A. LONGEST RUN ON A SNOWBOARD

Input:	Standard Input
Output:	Standard Output
Time Limit:	3 seconds
Memory Limit:	64 Megabytes

Michael likes snowboarding, since snowboarding is really fun. The bad thing of snowboarding is that in order to gain speed, the snowboard must slide downwards (from higher height to lower height). Another disadvantage is that after you have reached the bottom of the hill, you have to spend a lot of time to walk up again or wait for the ski-lift.

In order to maximize the fun, Michael would like to know how long the longest run an area is, so that he can play snowboarding in the area with the longest run. The area is given by a grid of numbers, defining the heights at various points in the area. For example, the following shows a 5×5 area.

```
 1  2  3  4  5
16 17 18 19  6
15 24 25 20  7
14 23 22 21  8
13 12 11 10  9
```

One can slide down from one point to another *connected* point if and only if the height decreases. One point is connected to another if it is at left, at right, above or below it on the area map. In the sample map, a possible slide would be 24-17-16-1 (start at 24, and end at 1). Of course if you would go 25-24-23-…-3-2-1, it would be a much longer run. In fact, it is the longest run possible.

INPUT

The first line contains the number of test cases T ($1 \leq T \leq 15$).

Each test case starts with a line containing the number of rows R and the number of columns C ($1 \leq R, C \leq 100$). After that follow R lines with C numbers each, defining the heights. The heights are always integers between 0 and 100.

OUTPUT

For each test case, print a line containing the length of the longest run one can slide down in that area.

EXAMPLE

Standard Input	Standard Output
2 10 5 56 14 51 58 88 26 94 24 39 41 24 16 8 51 51 76 72 77 43 10 38 50 59 84 81 5 23 37 71 77 96 10 93 53 82 94 15 96 69 9 74 0 62 38 96 37 54 55 82 38 5 5 1 2 3 4 5 16 17 18 19 6 15 24 25 20 7 14 23 22 21 8 13 12 11 10 9	7 25

Problem B. PACKAGE BOXES

Input:	Standard Input
Output:	Standard Output
Time Limit:	3 seconds
Memory Limit:	64 Megabytes

One day, Peter has to pick up a number of packages for his friend, Mary. There are N packages in total. Each package can be seen as a box with certain volume. A large box can hold another small box if and only if the volume of the large box is AT LEAST TWICE as large as the small box. A box with a small box inside cannot hold another small box. A box which has another box inside cannot be held by another box.

Peter can only take ONE package (or maybe TWO if the outer package has another package inside it) each time. Please find the minimal times that Peter needs to pick up all the packages.

.

INPUT

The first line contains an integer T ($1 \leq T \leq 10$) indicating the number of test cases.

For each test case, the first line contains an integer N ($1 \leq N \leq 10^5$). The second line contains N integers (each $\leq 10^7$) separated by spaces. The i-th integer indicates the volume of the i-th package.

OUTPUT

For each test case, output a line with an integer indicating the number of times Peter needs to pick in order to pick up all the packages.

EXAMPLE

Standard Input	Standard Output
3 4 2 1 5 8 4 2 3 4 8 8 2 5 7 6 9 8 4 2	2 2 5

(This is a blank page.)

Problem C. RUNNING PLAYER

Input:	Standard Input
Output:	Standard Output
Time Limit:	10 seconds
Memory Limit:	64 Megabytes

A city hunter game is divided into a number of distinct check points. A player has to start from the start check point, complete the required task in different check points, and arrive the finish check point within the required time.

Given the start check point, the finish check point, and the total number of check points from start to finish, you are required to find the total number of different routes. Note that it is possible for the player to repeat the check points they have visited during the game.

The game is described by a set of check points $\{c_1, c_2, \dots, c_n\}$, where $n \geq 1$. There are r routes directly connecting the check points. The routes in the game are described by a set of check point pairs, where each route is of the format (c_i, c_j), indicating that there is a direct route between check point c_i and check point c_j, where $1 \leq i \leq n$ and $1 \leq j \leq n$.

You are required to find the number of different paths to complete the game (i.e., reach the finish check point) with the restriction that the player must run through at most m routes, where $m \geq 2$.

For example, the path $c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow c_4 \rightarrow c_5$ means that the player starts at c_1, runs through 4 routes to finish at c_5.

As another example, the path $c_1 \rightarrow c_2 \rightarrow c_4 \rightarrow c_2$ means that the player starts at c_1, runs through 3 routes to finish at c_2.

As yet another example, the path $c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow c_2 \rightarrow c_5 \rightarrow c_6$ means that the player starts at c_1, runs through 5 routes to finish at c_6.

The player is not allowed to stop at a check point c_i unless there is a route called (c_i, c_i).

INPUT

Input comes from standard input. The format of each case is as follows. The first line is an integer n ($1 \leq n \leq 1000$) indicating the number of check points in the game. It is followed by an empty line, the number of routes r ($1 \leq r \leq 10^6$) between the check points, then the check point pairs describing the routes are listed line-by-line in the format c_i, c_j. Next is an empty line followed by the names of the starting check point and the finish check point in the format of c_s, c_f. Next line contains the limit m ($2 \leq m \leq 1000$) of routes that the player can run. Each test case is separated by two empty lines.

OUTPUT

For each test case, output a line which contains the number of possible paths to finish the game.

EXAMPLE

Standard Input	Standard Output
4 4 c_1,c_2 c_1,c_3 c_2,c_4 c_3,c_4 c_1,c_4 2 3 3 c_1,c_2 c_2,c_3 c_3,c_1 c_1,c_2 3	2 3

Problem D. RUN-LENGTH CODING

Input:	Standard Input
Output:	Standard Output
Time Limit:	3 seconds
Memory Limit:	64 Megabytes

Run-length coding is one of the widely used data compression methods. One of the run-length coding implementation is as follows:

- There are a total of 125 control characters with a run-length count, including:
    - Repeating control characters: $r2, r3, \dots, r63$:
        - Each ri, where $i = 2, 3, \dots, 63$, is followed by either another control character or a symbol. If the following symbol is another control character, ri (alone) signifies that the space character (i.e., blank) repeats i times, else, ri signifies that the symbol immediately after it repeats i times.
    - Non-repeating control characters: $n1, n2, \dots, n63$:
        - Each ni, where $i = 1, 2, \dots, 63$, is followed by a sequence of i non-repeating symbols.
- If a sequence of symbols of i $(i = 2, 3, \dots, 63)$ consecutive space is found, output a single control character ri.
- If a sequence of symbols of i $(i = 3, 4, \dots, 63)$ consecutive symbols other than spaces is found, output two characters: ri followed by the repeating symbol.
- Otherwise, identify a longest sequence of symbols of $i = 1, 2, \dots, 63$ non-repeating symbols, where there is no consecutive sequence of 2 spaces or of 3 other characters, and output the non-repeating control character ni followed by the sequence of non-repeating symbols.

Write a program to implement the above algorithm to code a symbol sequence. The input may consist of any letters, numbers, space or symbols.

INPUT

The first line indicates the number T $(1 \leq T \leq 100)$ of test cases.

Each test case contains a line of n symbols $(1 \leq n \leq 10^6)$ to be coded by the above method.

OUTPUT

Each line contains the coded sequence for each case.

EXAMPLE

Standard Input	Standard Output
3 AAAAABBBB A BBBB ABCD	r5Ar4B n1Ar2r4B n4ABCD

Problem E. COIN CHANGES

Input:	Standard Input
Output:	Standard Output
Time Limit:	3 seconds
Memory Limit:	64 Megabytes

Alice and Bob are good friends. They travel to Europe together. As we know, Euro is the official currency of the Eurozone. It has 1c (cent), 2c, 5c, 10c, 20c, 50c, 1EUR, 2EUR coins, and 5EUR, 10EUR, 20EUR, 50EUR, 100EUR, 200EUR, 500EUR banknotes. We don't care the differences between coins and banknotes, and we use cents to represent all values. In this way, there are 15 different values, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000.

Now, Alice and Bob have a banknote (or maybe coin) of value x, where x is one of the 15 values above. They need to buy some tickets with price y ($1 \leq y \leq x$) from an old machine. However, the machine is so old that it only accepts exact fare (i.e., you must put the exact amount of cash or the purchase is unsuccessful). Therefore, they decide to get the changes by buying some cheap things before buying the tickets.

However, they suddenly notice that they cannot predict how the change will be composed. For example, suppose that they have 50000, and the ticket price is 1. If they buy something of price 1, the change may be 1 × 5-cent + 24997 × 2-cent = 49999, and there is still no 1-cent for the ticket machine. Alice starts to calculate the minimum amount they need to spend so that the required fare is guaranteed in the change. In this example, she finds that, if only 1 buy is allowed, the buy should be 49997 so that the change (3 cents) is guaranteed to contain the 1-cent coin for the ticket.

On the other hand, Bob notices that they can buy multiple times. In this example, they may first buy something of price 1. The change may be 1 × 5-cent + 24997 × 2-cent = 49999, without a 1-cent coin. But, with a second buy of price 1 using a 2-cent coin, they will get the required 1-cent coin. In this case, only a total of 2 cents are spent.

Please help them to calculate the minimum amount they need to spend in Alice's way and that in Bob's way.

INPUT

The first line contains an integer T ($1 \leq T \leq 88888$) indicating the number of test cases.

Each test case contains only 1 line, storing 2 integers, x and y as discussed before.

OUTPUT

For each test case, output a line with two integers, indicating the amount that would be spent in Alice' way and that in Bob's way.

EXAMPLE

Standard Input	Standard Output
3 50000 1 10 2 1 1	49997 2 1 1 0 0

Problem F. WORD BLOCKS

Input:	Standard Input
Output:	Standard Output
Time Limit:	10 seconds
Memory Limit:	64 Megabytes

Given a word puzzle of size $x \times y$ and a target word block of size $m \times n$, write a program to output the position x_0, y_0 of the top left corner of the target word block on the word puzzle. Note that the word block can be rotated by 90, 180, or 270 degrees anticlockwise.

INPUT

For each test case, the first line is the dimension of the crossword puzzle $\times y$ $(1 \leq x, y \leq 1000)$, followed by y lines of content. Each of these lines contains x letters, separated by a blank space. It is the followed by a blank line, then the dimension of the target word block $m \times n$ $(1 \leq m, n \leq 1000)$, followed by n lines of content. Each of these lines contains m letters, separated by a blank space.

Every test case is separated by an empty line.

OUTPUT

Your output should be the location (x_0, y_0) of the top-left corner of the puzzle block that matches the target word block. Top-left corner of the puzzle is (0,0), and x increases towards right and y increases towards down. If there are more than one solution, output all of them, one in each line, sorted by x then y.

If there is no such word block, output the text "No match".

EXAMPLE

Standard Input	Standard Output
10 10 a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a b a a a a a a a a c d a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a 2 2 a b c d 10 10 a a a a a a a a a a a a a c a a a a a a a a a d b a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a b a a a a a a a a c d a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a 2 2 a b c d	4 5 3 1 4 5

Problem G. CITY PLANNING

Input:	Standard Input
Output:	Standard Output
Time Limit:	3 seconds
Memory Limit:	64 Megabytes

Blink comes to a fantasy world with N cities. There are some roads across which people can travel between the cities. Now the king asks Blink to build as few as possible roads so that there is at least one pathway exists between any two cities. The king wants to know how many such plans exist in total. You may assume that we may build a road to connect any pairs of cities if required.

INPUT

The first line contains an integer T $(1 \leq T \leq 10)$ indicating the number of test cases.

For each test case, the first line contains three integers, N $(2 \leq N \leq 10^5)$, M $(0 \leq M \leq 10^5)$ and P $(1 \leq P \leq 10^9)$, which means that there are N cities and there M roads exist connecting the cities. Afterwards, there are M lines of integer pairs u_i v_i indicating a road between city u_i and city v_i.

OUTPUT

For each test case, output a line with an integer indicating the total number of possible plans, mod P.

EXAMPLE

Standard Input	Standard Output
10 7 100000 1 2 1 3 2 3 4 6 5 6 7 8 9 10 4 1 100000 1 2	3600 8

— End of Problem Set —

(This is a blank page.)