The 2021 ICPC Asia Topi Regional On-site Contest

Instructions

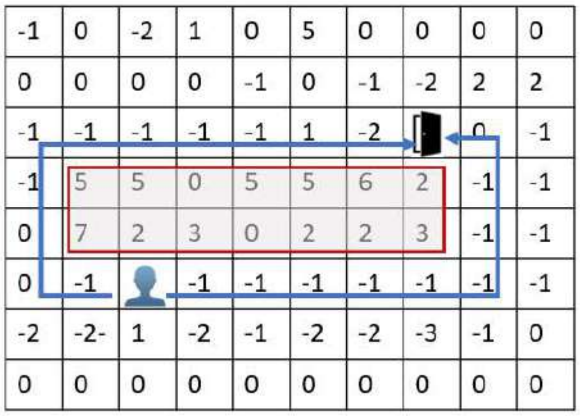
- Do not open the booklet unless you are explicitly told to do so. You can only read these instructions below.
- Do not create disturbance or move around unnecessarily in the arena.
- If you have any question regarding the problems, send a clarification from the judges using DOMJudge.
- There would be no internet access and mobile phones are also not allowed.
- Before submitting a run, make sure that it is executable via command line. For Java, it must be executable via "javac" and for GNU C++ via "g++". Java programmers need to remove any "package" statements and source code's file name must be the same as of main class. C++ programmers need to remove any getch() / system("pause") like statements.
- Do not attach input files while submitting a run, only submit/attach source code files, i.e., *.java or *.cpp or *.py.
- Language supported: C/C++, Java and Python3
- Source code file name should not contain white space or special characters.
- You must take input from Console i.e.: Standard Input Stream (stdin in C, cin in C++, System.in in Java, stdin in Python)
- You must print your output to Console i.e.: Standard Output Stream (stdout in C, cout in C++, System.out in Java)
- Please, don't create/open any file for input or output.
- Please strictly meet the output format requirements as described in problem statements, because your program will be auto judged by computer. Your output will be compared with judge's output byte-by-byte and not tolerate even a difference of single byte. So, be aware! **Pay special attention to spaces, commas, dots, newlines, decimal places, case sensitivity etc.**
- Unless mentioned in some problem, all your programs must meet the time constraint of 5 seconds.
- The decision of judges will be absolutely final.

PROBLEM 1: Alien obstacle

Time limit: 3 seconds

You suddenly wake up on an alien land where little green men are trying to kidnap you. You soon realize that you're in a 2D grid, where each cell is assigned a number and there is only one door from which to escape. Your job is to find if it is possible to escape and the number of steps that may be required to make it to the door.

Those clever little green men, however, are making this job difficult for you. They can very quickly block your way be erecting barricades on different cells in the grid. The barricades, however, have to be constructed on consecutive cells. You also realize that the numbers associated with each cell indicates the cost-benefit of putting a barricade on the cell. The little green men being greedy only block those cells that will result in maximum benefit for them. An example is shown in Fig. 1. below:



-1	0	-2	1	0	5	0	0	0	0
0	0	0	0	-1	0	-1	-2	2	2
-1	-1	-1	-1	-1	1	-2		0	-1
-1	5	5	0	5	5	6	2	-1	-1
0	7	2	3	0	2	2	3	-1	-1
0	-1		-1	-1	-1	-1	-1	-1	-1
-2	-2-	1	-2	-1	-2	-2	-3	-1	0
0	0	0	0	0	0	0	0	0	0

Fig. 1. Two possible routes to escape after blockages placed by little green men.

Note that the blockage must be made up of consecutive rows and columns and cannot take irregular shapes. Your job is to find the blockage the greedy little green men will place and figure out if there is a way out or not. If there is, then you must know the minimum number of steps to be taken. In the figure above, the first path (going left) requires 12 steps while the second path (going right) requires 10 steps.

The grid is labeled from the top left. In the example above, the top left is (1,1) while the bottom right is (8,10).

Input

The first line of input contains an integer 'T', $1 \leq T \leq 10$, which denotes the number of test cases. Each subsequent line is a test case and contains the number of rows $1 \leq R \leq 1000$, number of columns $1 \leq C \leq 1000$, your position given as (X,Y), the door position given as (X,Y), followed by a row-wise entry of each element. All elements are comma separated.

Output

For each test case, print the following (in a single line)

1. Print 1 if it is possible to reach the exit and 0 otherwise.
2. The starting position (X_1,Y_1) of the blockade

3. The ending position (X_2,Y_2) of the blockade
4. The length of path required to the door (or -1 if no path exists)

The values are comma separated with a space between different entries (but not within an entry) as shown in sample input/output.

Sample input & output

The following is an example of a sample input and corresponding correct outputs. The second test case corresponds to the example given in Fig. 1. above. (Note that the 8×10 matrix shown here is wrapped but entered as a single row).

Sample input	**Sample Output**
2 3, 3, (3,1), (1,2), -1, -1, -1, 1, 1, 1, -1, -1, -1 8, 10, (6,3), (3,8), -1, 0, -2, 1, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, -1, -2, 2, 2, -1, -1, -1, -1, -1, 1, -2, 0, 0, -1, -1, 5, 5, 0, 5, 5, 6, 2, -1, -1, 0, 7, 2, 3, 0, 2, 2, 3, -1, -1, 0, -1, 0, -1, -1, -1, -1, -1, -1, -1, -2, -2, 1, -2, -1, -2, -2, -3, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, (2,1), (2,3), -1 1, (4,2), (5,8), 10

PROBLEM 2: Happy Customers

Time limit: 5 seconds

In wake of high demand of dates in the holy month of Ramadan, the Sabeel Cash and Carry introduced a new packaging scheme for customers on discounted prices. The dates will be available in two different packages of size ***A*** and size ***B***. The sizes ***A*** and ***B*** are decided by staff as per the demand. However, there are limited stock available for each day.

Given the size of the packages ***A*** and ***B***, the total quantity of dates stock ***D*** and the number of customers ***N***, find the maximum number of customers that can be served with the given dates stock in a single day.

You should display the maximum number of customers that can be served along with their index. Assume the index of customers starts from 1.

Input

The first line of input contains two integers ***N*** *(no. of customers)* and ***D*** *(total stock available)*; next line contains two integers denoting the sizes of packages ***A*** and ***B**, respectively*. The next ***N*** lines contain two integers for each customer denoting total number of packages of size ***A*** and size ***B*** (separated by a space) that customer requires; where ***N*** <= ***100***.

Output

Print maximum number of customers that can be served and in next line print the space separated indexes of served customers in serving order. Please note that the customer with minimum servable order will be printed first, followed by the second minimum servable order, and so on. In case of similar sized order, the customer with the lower index will be printed first.

Sample input & output

Sample input	**Sample Output**

6 4
1 1
2 0
3 2
4 4
10 0
0 1
4 2
	2
	5 1

PROBLEM 3: Hopping Bunnies

Time limit: 3 seconds

Bunnyland is a magical place that is found in a galaxy far, far away. Bunnies that are living on one of the planets in that galaxy are very energetic and restless. They are constantly hopping on different things. So, their master entertainer created a game for the hopping bunnies that will be used to consume their energy as well as award them a prize in the form of carrots. Obviously, each bunny would try to get as many prizes by playing the game as possible. The game is played as under.

The arena for the game is in the form of a square that contains 09 jumping pads of different colors on which the bunnies would jump. Of these 09 jumping pads, only Red, Green, Blue and Yellow pads have significance in the form of getting some prize. The rest will be useless, i.e., only jumping on the 04 above-mentioned pads can get them a prize of carrots under following rules. The distance between these jumping pads does not have any significance.

1. Jumping on Red pad will straight away result in the award of as many carrots as the number of jumps.
2. Yellow pad has magical properties, and it could only work in combination with Red, Blue and Green pad. Jumping individually on Yellow, Blue or Green pads would not award any carrot. Jumping on Yellow followed by Red, Green or Blue would be counted as one jump.
3. If a bunny hops on Yellow pad and then Red pad, it can select the total number of carrots that a bunny has been awarded so far and can re-produce the exact number of carrots awarded as explained in the following 2 steps. It will only work if immediately before this jumping sequence, carrot(s) have already been awarded.
4. If a bunny hops on Yellow pad and then Blue pad, the imaginary carrot(s) (obtained in step 3) are ensured and these carrots can be obtained later as explained in step 5.
5. If a bunny hops on Yellow pad and then Green pad (provided step 4 is already done), it will be awarded as many carrots as it has obtained before starting step 3. Repetition of this step will result in the award of carrots again and again.
6. Bunnies can jump on pads for a certain number of time that the master entertainer will inform them before starting the game play.

Bunnies would like to win maximum number of carrots by jumping certain number of times on the 4 colored pads. Write a program to compute this maximum number given the number of jumps.

Example:

Input: N = 7
Output: 9
Bunny can get at most 9 carrots by jumping in the following sequence.
Red, Red, Red, Yellow-Red, Yellow-Blue, Yellow-Green, Yellow-Green.

Input

The first line of the input is the number of test cases, **T**. The next **T** lines represent each of the test cases and contain a single integer, **N**, the number of jumps that a bunny is allowed.

Output

The output also contains **T** entries, one for each test case. For each test case, print the maximum number of carrots that a bunny can win.

Constraints

$0 < \mathbf{N} < 75$

Sample input & output

Following is a sample input and corresponding correct output.

Sample input	Sample Output
3 6 7 9	6 9 16

PROBLEM 4: Faulty AutoWaiter Drone

Time limit: 3 seconds

The “*Aerial Robotics and Vision Lab*” at GIK Institute has developed an in-house drone “*Automatic Waitedrone*” that can perform all duties of a waiter including delivering food and liquids and picking up utensils from the table. The lab has also developed a customized version of the robot called “*Circular AutoWaitor*” that can automatically determine the number of participants sitting in a circle and serve them.

The lab has donated a *Circular AutoWaitor* to ICPC Pakistan Chapter management, who are using this aerial drone to serve participants in the annual dinner. The participants are seated on a table in a circular manner. However, due to a technical fault, the drone does not pick all the utensils in a sequential order after the dinner. Instead, it will pick up utensils from the first person, skip one person, pick up utensils from the third person, skip the next two persons, and so on until all seats are cleared. Once the utensils from a seat have been picked, the drone does not include the cleared seat in the skip count. Since the participants are sitting in a circle, the process of picking up utensils stops when all the seats are cleared.

For example, if 5 participants are seated, then the utensils will be picked up in the following order – 1 3 2 5 4, and the utensils are taken last from seat number 4.

Similarly, if 4 participants are seated, then the utensils will be picked up in the following order – 1 3 4 2, and the utensils are taken last from seat number 2.

Given the number of people sitting at the table, can you help ICPC identify the seat from which the Circular AutoWaiter will pick up the utensils last?

Input

The first line gives the number of test cases n ($0 < n < 1000$), which is followed by n lines for each test case. Each test case contains a single entry – the total number of people p ($0 < p < 5000$) sitting in the round table.

Output

The output of the program contains n ($0 < n < 1000$) lines, one for each test case, indicating the seat from which the utensils will be picked up last.

Sample input & output

Following is a sample input and corresponding correct output.

Sample input	Sample Output
5	4
5	4
7	8
12	11
15	13
25	

PROBLEM 5: Order of nature

Time limit: 3 seconds

Digits can show connections, connections between you, me and our world. that will make you see the world in a whole new way. In the late 1800s, in a logarithm book someone noticed that the very first pages of the book were worn out and the last pages were untouched. This was also the case with nearly all other books in the library. It was a first clue to a profound and secret order to the world all around us. one that could pop up in everything from your favorite sport to your favorite song even to the way you're going to die. once you see it you can't unsee it. it's everywhere.

The observation suggests that in many real-life sets of numerical data, the leading digit is likely to be small. Leading digit as we know is the first digit in any number. So, number 1 appears as the leading digit about 30 % of the time while 9 appears less than 5 % of the time. It has been shown that this result applies to a wide variety of data sets, including electricity bills, street addresses, stock prices, house prices, population numbers, death rates, lengths of rivers, and physical and mathematical constants.

A set of numbers is said to satisfy this natural order if the leading digit d (d ∈ {1, … , 9}) occurs with probability

$$P(d) = \log_{10}(d+1) - \log_{10}(d) = \log_{10}\left(\frac{d+1}{d}\right) = \log_{10}\left(1+\frac{1}{d}\right)$$

It gives P(1)=30.1 % and P(2)=17.6 % as occurrences of 1 and 2 as leading digits. You can calculate the rest upto 9 using the above formula. In an example, examining a list of the heights of the 58 tallest structures in the world shows that 1 is by far the most common leading digit in those heights, irrespective of the unit of measurement (meters or feet), with count of 24 and occurrence of 41.3%, (as 24 is 41.3% of 58) as shown in sample input 1. That is 24 building's heights had 1 as their leading digit. So you see occurrence(1) differs from the natural order P(1) by 11.2% (absolute difference). Your job is to find how much a set obeys the natural order. Output the sum of differences of occurrences of leading digits from natural order.

Sum (abs(P(d) - Occurrence(d))) where d = 1 to 9

Input

The input consists of an integer *T* representing numbers of test cases. For each test case, there is a single line with nine non-negative integers representing counts of leading digits (1 to 9) for a certain system.

Output

Find occurrences from the given counts and output a number showing deviation from natural order. Each output must be on a separate line. An error of <=0.001 will be tolerated and output will be accepted.

Sample input & output

Following is a sample input and corresponding correct output.

Sample input	Sample Output

2
24 9 7 6 1 5 1 4 1
29 17 12 10 7 6 5 5 5
	31.27489
	3.32399

PROBLEM 6: Tree path sum

Time limit: 5 seconds

In this problem, you are required to verify whether a path of given length exists in the given binary tree. The length of the path will be dependent on the weights of the nodes. Given below are lengths of a few paths from the graph given in Fig. 2.

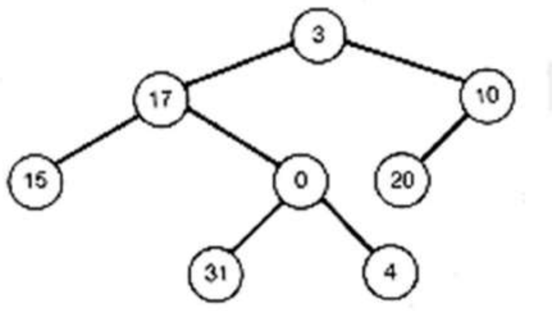


Fig. 2. An example of a graph with weights on the nodes.

```
len(3,15)=35 (given as 3+17+15)
len(3,20)=33 (given by 3+10+20)
len(20,20)=20 (given by 20)
```

A binary tree can be constructed if any two of the traversals are provided, i.e., (pre-order, post-order, in-order). You will be provided two traversals of a binary tree, i.e., pre-order and post-order. You are required to compute whether a path exist between any two nodes of the tree for the given length.

Input

The first line will contain the number of test cases, *N*. Each of the test case will contain 3 lines. The first line will represent a tree in in-order fashion, the second line will contain the post-order representation of the tree. The third line of a test case contains *T* tests where each test represents weight of a path to be checked in the tree.

Output:

You are required to give *T* outputs for each test case (*T* may vary for each test case). Each output will contain either TRUE or FALSE on separate lines for each of the integers provided in the third line of each test case. For instance, the first FALSE in the output denotes that there exists no path of `len=25` and the last TRUE corresponds to `len=35`.

Sample input & output

Sample input	Sample Output

1
15 17 31 0 4 3 20 10
15 31 4 0 17 20 10 3
25 12 33 20 8 9 1 35
	FALSE
	FALSE
	TRUE
	TRUE
	FALSE
	FALSE
	FALSE
	TRUE

PROBLEM 7: Car rental

Time limit: 4 seconds

Ali owns a car rental company, and his business is booming. The company has *k* drivers, each in charge of one set of cars. Every week, the company accepts *n* rental requests. For every request, they send a driver with the car to the location. The driver delivers the car, sets up the meter, and instructs them on how to use the car. After the trip, the person who asked for the rental is responsible for returning the car back to Ali's company.

Unfortunately, in some weeks the number of drivers is less than the number of requests, so some drivers may have to be used for more than one request. In such cases, the company cannot wait for the person to return the car and must keep the driver with the car to drive to another location. The company has an accurate estimate of the cost to drive a car from any location to any other location.

Given these costs, Ali wants to prepare an Advance Car Rental Map to service the requests while minimizing the total driving cost of car (including the cost of the first move), even if that means not using all the available drivers. Ali needs your help to write a program to accomplish this task. The requests are sorted in ascending order of their car delivery times and they are chosen in such a way that for any $i < j$, there is enough time to drive the car used in the i^{th} request to the location of the j^{th} request.

Input

The first line of the input contains the total number of test cases. The subsequent lines contain input data for the test cases as follows. The first line of each test case contains two integers n $(1 \leq n \leq 100)$ and k $(1 \leq k \leq 100)$ which are the number of requests and the number of drivers, respectively. Following that are n lines, where the i^{th} line contains $n-i+1$ integers between 0 and 1 000 000 inclusive. The j^{th} number in the i^{th} line is the cost of driving a car from location i to location $i+j$. The company is at location 1 and the n requests are at locations 2 to $n+1$. (Note that lines i and j are relative to the test case and not the entire input file).

Output

The output should display one line for each test case. The output of each test case is the minimum moving cost to service all rental requests. (This amount does not include the cost of driving the car back to the rental company.)

Sample input & output

Following is a sample input and corresponding correct output.

Sample input	Sample Output

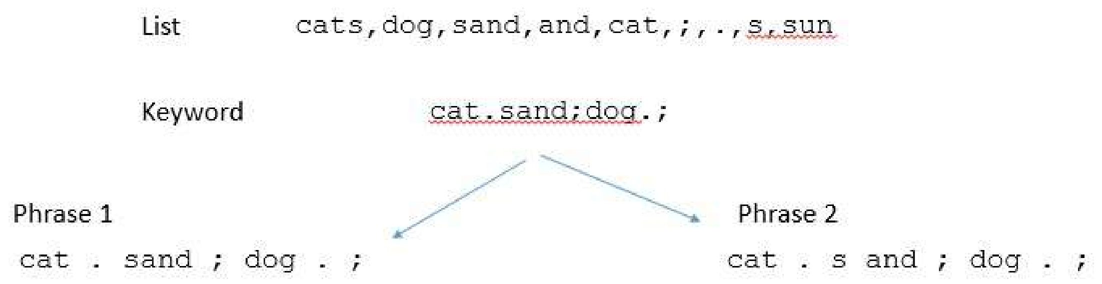
2
3 2
40 30 40
50 10
50
3 2
10 10 10
20 21
21
	80
	40

PROBLEM 8: Phrases

Time limit: 3 seconds

AbC is a very forgetful person. Unfortunately, he is in the era of internet and social media where all the websites require a unique phrase. The biggest challenge he faces in using these websites, is remembering these unique phrases. After much effort, he devised a mechanism to remember them. He made a list of items that he can use in generating a phrase. But this proved to be more challenging than he thought as he must figure out a way to generate a phrase. One of his friends gave him an idea of using a keyword that would help him find the correct phrase. He has yet to make such a keyword, but he is confused that how many phrases he would be able to generate from a keyword with a list. A phrase is selected only when a keyword can be split into words from the list while maintaining the order of elements of keyword. If a keyword contains any word/alphabet that is not in the list, that keyword cannot be used for a phrase. Please help him in finding maximum number of phrases that are possible from a keyword using a list of words.

Example



Note that all the elements of a phrase are present in the list and the order of elements is same as in the keyword. Therefore, the output is 2.

Input

The first line of the input consists of *t*, (1<=*t*<=100) representing the total number of test cases. For each test case, there are two lines. The first line is a list of comma-separated values is provided in a single line. The next line contains the keyword.

Output

Output consists of *t* lines, each line contains exactly one number representing the number of phrases that can be generated.

Sample input & output

Following is a sample input and corresponding correct output.

Sample input	Sample Output
4 cats,dog,sand,and,cat,;,.,s,sun catsandog.; cats,dog,sand,and,cat,;,.,s,sun catssanddog.; cats,dog,sand,and,cat,;,.,s,sun cat.sand;dog	0 4 2 2

cats,dog,sand,and,cat,;,.,s,sun cat.sand;dog.;cats,dog,sand,and,cat,;,.,s,sun cat.sand;dog.;	

PROBLEM 9: Exam seating plan

Time limit: 5 seconds

In this task you are going to help the examination department of a university in their seating planning during exams. The examination department is required to plan a seating in such a way that the neighbor of one student should not be the one who will take same Exam. In the figure given below the nodes represents the seating arrangement of 6 students in an examination hall. The edge represents the students' seats where they fall in the category of neighbors, and we need to have student of another course. At the same time the other requirement is to use minimum number of courses (optimize) in one Exam Hall as that will make the arrangements much easy.

For example, consider the following seating plan of 6 students shown in figure given below in one Hall. The edges show the neighborhood relation between the students. The color shows the paper (P) to be distributed. In the case given below the minimum number of papers that can be managed in Hall is 3 as shown with yellow (ST1, ST5), green (ST2, ST4, ST6), and red (ST3) colors, while following the rule of having no similar exam paper of neighbors.

Note:

- The maximum number of students seating is 100 in any eximination Hall.
- In case of having no seats (vertices), zero number of papers should be returned.
- Start node numbers from 1.

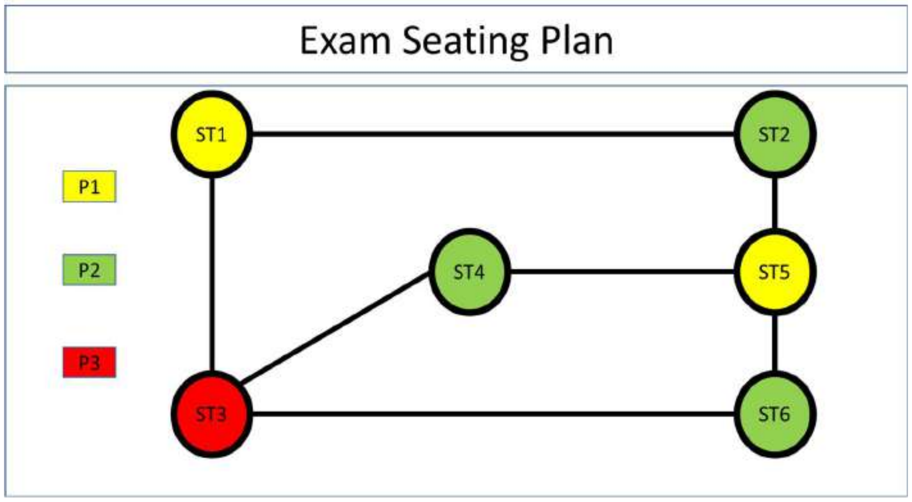


Fig. 4. Exam seating plan.

Input

The first line should contain the number of test cases. This is subsequently followed by the test cases. The first line of each test case will contain two integers: S and E(edges) representing the number of students in the exam hall and the number of neighborhood relationship, respectively. E.g., in the example given, 6 represents student seats capacity and 7 represents neighbor relations, which are single space seperated. This will be followed by E number of lines. Each E line will contain two integers C1 and C2. Here, C1 and C2 are neighbors. For the test case given, the first of the E lines show that ST1 and ST2 are neighbors.

Output

The output contains just one line for each test case. Each output will have one number which is the minimum number of exams (papers) that can be adjusted in one Exam Hall with the rule that no neighbors have same exam paper. E.g., in the given case, the answer will be: **3**

Sample input & output

Following is a sample input and corresponding correct output.

Sample input	**Sample Output**

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

PROBLEM 10: Research

Time limit: 10 seconds

Amir is researching computer vision. He works with grayscale images which are denoted as 2D matrices of integers. Each cell or pixel of the image matrix has a positive integer in range [0,255] in it. Where 0 denotes that the pixel is BLACK and 255 means that the pixel is WHITE, and values in between are shades of gray. For each picture, Amir runs a bunch of queries.

In a query, Amir calculates the bitwise-XOR of some rectangular submatrix in the image. A query will have four inputs: x_1, y_1, x_2, y_2, where (x_1, y_1) denotes the top-left corner of the sub-matrix and (x_2, y_2) denotes the bottom-right corner of the sub-matrix. Values for x and y are 0-based. All the values in the sub-matrix are bitwise-XORed together, and the result is the answer for the query.

For example, let's assume Amir has the following picture:

0	0	192	0	0	16	0
12	0	24	0	48	255	96
0	24	0	0	64	128	0
0	0	48	0	128	0	0
48	0	0	24	0	0	255
0	0	255	0	48	0	0
0	0	0	192	0	12	32

If Amir queries $(x_1, y_1, x_2, y_2) = (4, 1, 6, 2)$, then the sub-matrix will be the highlighted rectangle in the above table, and the answer for the query will be ($\oplus$ is the symbol for XOR):

$$48 \oplus 255 \oplus 96 \oplus 64 \oplus 128 \oplus 0 = \mathbf{111}$$

Hence the result for the query is 111. Amir hired your help in his research. You will be given an image and multiple queries. For each query, you must calculate the answer and print it. Refer to the Input and Output format Amir shared with you below.

Input

First line consists of 2 integers H and W, denoting the dimension of the image (H x W). Next H lines contain W integers each, where each integer is the greyscale value in range [0-255].

The next line contains a single integer Q, denoting the number of queries. The next Q lines contain 4 integers each, the values for x_1, y_1, x_2, y_2.

Output

For each query, you have to print a single line in the format: Query #i: x

where *i* denotes the query number (1-based sequence) and *x* is the answer for the query calculated by XOR-ing all the values in the rectangle denoted by x_1, y_1, x_2, y_2.

Limits

$1 \leq \mathbf{H}, \mathbf{W} \leq 350$

$0 \leq \mathbf{x_1} \leq \mathbf{x_2} < \mathbf{W}$

$0 \leq \mathbf{y_1} \leq \mathbf{y_2} < \mathbf{H}$

$1 \leq \mathbf{Q} \leq 10^4$

Sample input & output

Following is a sample input and corresponding correct output.

Sample Input	Sample Output

7 7
0 0 192 0 0 16 0
12 0 24 0 48 255 96
0 24 0 0 64 128 0
0 0 48 0 128 0 0
48 0 0 24 0 0 255
0 0 255 0 48 0 0
0 0 0 192 0 12 32
5
4 1 6 2
6 6 6 6
2 3 3 4
0 1 1 2
0 0 6 6
	Query #1: 111
	Query #2: 32
	Query #3: 40
	Query #4: 20
	Query #5: 247