	Problem A Flying Squirrel Time Limit 1 sec	2018 ICPC Asia Nakhon Pathom Regional Contest

There are N poles of height $H_1, H_2, H_3, \dots, H_N$ (for all i, $H_i > 0$). An infamous flying squirrel, started at position K, can glide to any lower pole on the left or on the right, as described by the conditions below.

1. He can glide to a lower pole on the left, to position L say, if $0 < H_L < H_K$ (where $1 \le L < K$) and $H_i < H_K$ for all i with $L < i < K$.
2. He can also glide to a lower pole on the right, to position R say, if $0 < H_R < H_K$ (where $K < R \le N$) and $H_i < H_K$ for all i with $K < i < R$.

To be brief, the squirrel cannot pass a pole whose height is higher than or equal to the height of his starting pole. Moreover, the squirrel can only stop on the top surface of a pole. He cannot hang from the side.

The flying squirrel wants to demonstrate how to glide to his younger fellows. Therefore, he wants you to find a longest path that he can glide from his starting location.

Input:

1st line: Receive two integers N and M which represent the number of poles and the number of queries, respectively ($1 \le N, M \le 100{,}000$).

2nd line: Receive N numbers $H_1, H_2, H_3, \dots, H_N$, where H_i represents the height of pole at position i ($1 \le H_i \le N$).

Next M lines: Each line is corresponding to an information of a query. Each query contains two integers X and Y. X represents the starting location of the flying squirrel. Y represents the ending location of the flying squirrel. There are two cases:

- If Y > 0, your task is to find the maximum number of glides that the squirrel can make beginning at the pole at X and ending at the pole Y, or going the other way from Y to X.
- If Y = 0, you task is to find the maximum number of glides starting at the pole at X and ending at any position.

Output:

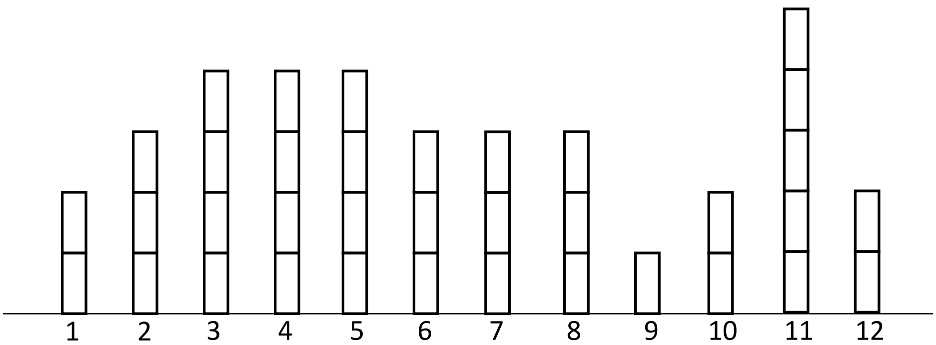
Return M lines that correspond to the length of the longest path of each query. Answer 0 if there is no glide path from position X to position Y (when Y > 0), or the squirrel cannot glide to any lower position (when Y = 0).

Sample Input/Output

Input	Output
12 8 2 3 4 4 4 3 3 3 1 2 5 2 3 0 4 0 5 0 7 0 7 11 7 9 11 1 9 12	2 0 3 0 2 0 3 0

Explanation:

All 12 poles in this example can be shown as the figure below.

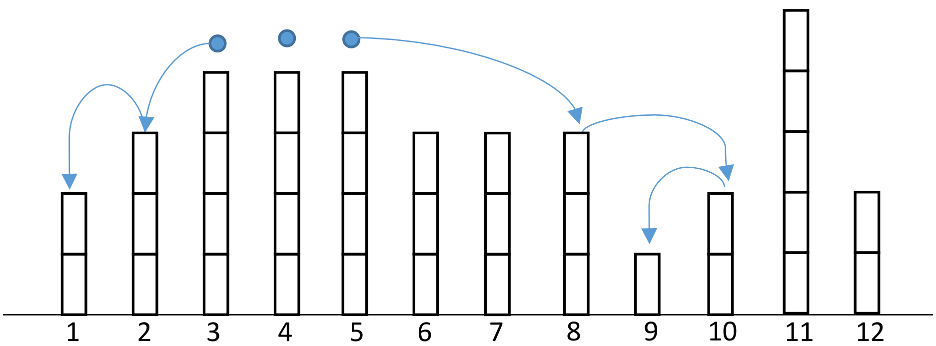


When Y = 0:

If X=3 and Y=0, the answer is 2 because he can glide from position 3 -> 2 -> 1.

If X=4 and Y=0, the answer is 0 because he cannot glide to anywhere lower ($H_3 = H_4 = H_5$).

If X=5 and Y=0, the answer is 3 because he can glide from position 5 -> 8 -> 10 -> 9.

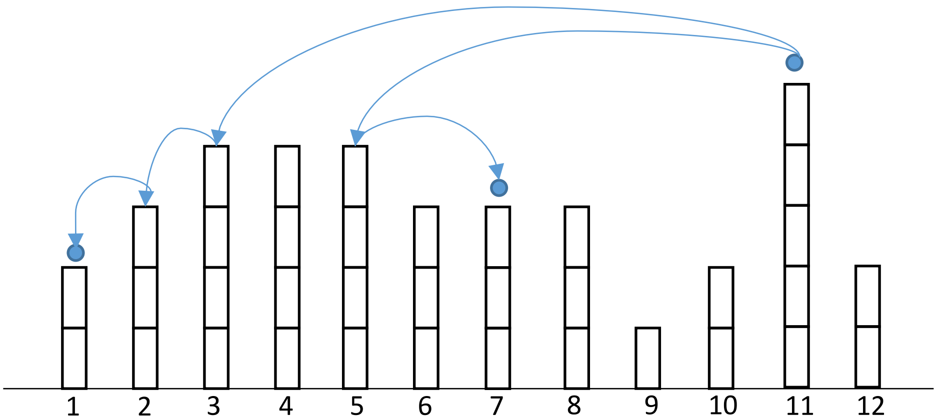


When Y > 0:

If X=7 and Y=11, the answer is 2 because the squirrel can glide from position 11 -> 5 -> 7.

If X=7 and Y=9, the answer is 0. He cannot glide from 7 -> 9 nor 9 -> 7 ($H_7 = H_8 > H_9$).

If X=11 and Y=1, the answer is 3 because he can glide from position 11 -> 3 -> 2 -> 1.


1
2
3
4
5
6
7
8
9
10
11
12

You are given a grid of R rows and C columns, so the grid has R * C cells. The grid looks very boring, so you decide to mark some of the cells. To mark one cell you need to follow the procedure below.

Pick K consecutive cells such that this exact group of cells hasn't been selected before. The cells must be either in the same row or in the same column. This group of K cells must have at least one unmarked cell.
Select one of the unmarked cells inside that group, and mark it with a red circle.

You want to mark as many cells as possible, so you need to figure out an optimal way of selecting the groups and cells.

However, after you finished coloring the grid, you got even more curious. Now you want to know how many different final states of the grid are possible which have the maximum number of cells colored. This number may be too large, so you need to print modulo 1000000007.

Input:

The first line of input contains T (1 <= T <= 50), the number of different tests. Each test is given in a separate line with three integers R, C, K (1 <= R, C <= 1000, 2 <= K <= 4).

Output:

For each test case you need to print one line in the format "Case id: result", where id is the test number and result is the number of different grids you can end up with that have the maximum number of cells colored.

Sample Input/Output

Input	Output
2 2 3 3 3 3 3	Case 1: 9 Case 2: 78

Explanation:
In the first case there are 2 rows and 3 columns. You have to take three consecutive cells. Since there is no way to take three consecutive cells in the same column as there are only two rows, we have options to select two triplets, one for each row. For each triplet we can mark one cell out of three, so overall there will be 9 possible different final states of the grid.

In the fantasy world of ICPC there are magical beasts. As they grow, these beasts can change form, and every time they do they become more powerful. A beast cannot change form completely arbitrarily though. In each form a beast has **n** eyes and **k** horns, and these affect the changes it can make.

A beast can only change to a form with more horns than it currently has.
A beast can only change to a form that has a difference of at most **w** eyes. So, if the beast currently has **n** eyes it can change to a form with eyes in range [**n** - **w**, **n** + **w**].

A beast has one form for every number of eyes between 1 and **N**, and these forms will also have an associated number of horns. A beast can be born in any form. The question is, how powerful can one of these beasts become? In other words, how many times can a beast change form before it runs out of possibilities?

Input

The first line contains two integers, **N** and **w,** that indicate, respectively, the maximum eye number, and the maximum eye difference allowed in a change ($1 \leq \mathbf{N} \leq 5000$; $0 \leq \mathbf{w} \leq \mathbf{N}$). The next line contains **N** integers which represent the number of horns in each form. I.e. the ith number, $h(i)$, is the number of horns the form with i eyes has ($1 \leq h(i) \leq 1\ 000\ 000$).

Output

For each test case, display one line containing the maximum possible number of changes.

Sample Input/Output

Input	**Output**	**explanation**
5 5 5 3 2 1 4	4	Start with 1 horn and 4 eyes, and it can change 4 times: (1 horn 4 eyes) -> (2 horns 3 eyes) -> (3 horns 2 eyes) -> (4 horns 5 eyes) -> (5 horns 1 eye).
5 1 5 3 2 1 4	3	Start with 1 horn and change so the number of horns goes 1 -> 2 -> 3 -> 5. Note that the beast can't go from 3 horns to 4 horns as the difference in the number of eyes is too big.
5 5 2 2 2 2 2	0	There is no possible change from any starting form.

In a rural village in Thailand, there is a long, straight, road with houses scattered along it. (We can picture the road as a long line segment, with an eastern endpoint and a western endpoint.) The Thai government is planning to set up bus stops on this road in such a way that every house is within ten kilometers of one of the stops. What is the minimum number of stops the government need to set up to satisfy the requirement?

Input:
The first line contains a single integer m representing the number of the test cases. Each test case consists of the following two lines:
The first line contains a single integer n representing the number of houses on the road, where $0 \leq n \leq 2{,}000{,}000$.
The second line contains n integers $h_1\ h_2\ \ldots\ h_n$ representing the locations of the houses in kilometers as measured from the start of the road, where $0 \leq h_i \leq 90{,}000{,}000$. That is, the first house is h_1 kilometers away from the start of the road, the second house is h_2 kilometers away from the start of the road, and so on. You may assume that h_i's are sorted in ascending order, e.g., $h_1 \leq h_2 \leq h_3 \leq \ldots \leq h_n$.

Output:
For each test case, print out in a single line the minimum number of bus stops that satisfy the requirement.

Sample Input/Output

Input	Output
2 5 1 2 3 200 210 4 10 30 80 200	2 3

You are given an array of **n** integers. Define a binary relation ~ on the elements of this array so that x ~ y if and only if |x – y| is less than or equal to 2. Let ~* be the transitive closure of ~. So x ~* y if and only if there is a sequence y_1, y_2, …, y_k = y with x ~ y_1 ~ y_2 ~ … ~ y_k. We say two numbers are in the same group if they are in the same ~* equivalence class.

For example consider this array: [2, 1, 7, 4, 3, 8, 9, 12, 13, 20]

This array has 4 groups:

[2,1,4,3]
[7,8,9]
[12, 13]
[20]

Your target is to transform the array so that the size of the largest group is maximized. To transform the array, you can make at most two moves. In each move, you can select an index and add or subtract 1 to/from the number at that index. It's not allowed to select the same index twice.

In the example above, you can transform the array into [2, 1, 7, **5,** 3, 8, 9, **11,** 13, 20] by adding 1 to the 4th index and subtracting 1 from the 8th index. Now the groups are:

[1, 2, 3, 5, 7, 8, 9, 11, 13, 20]
[20]

Size of the largest group is now 9.

Find the largest possible group size obtainable by transformation and print it.

Input:

The first line contains T (1 <= T <= 13000), denoting the number of test cases. The first line of each test case contains an integer **n** (1 <= n <= 100) denoting the size of the array. The second line contains n integers denoting the elements of the array. Each element will be between 1 and 200.

The input file is big, fast I/O is recommended.

Output:

For each test case, print the case number and the size of the largest possible group after transformation.

Sample Input/Output

Input	Output
5 10 2 1 7 4 3 8 9 12 13 20 3 1 2 3 4 1 2 7 8 3 10 30 20 2 5 5	Case 1: 9 Case 2: 3 Case 3: 2 Case 4: 1 Case 5: 2

regionals icpc International Collegiate Programming Contest icpc.foundation	Problem F Lucky Pascal Triangle Time Limit 1 sec	2018 ICPC Asia Nakhon Pathom Regional Contest

Do you know what a Pascal Triangle is? No? Need a refresher? Then here are the key points:

- Pascal Triangle is made of multiple rows, starting from 0.
- The i-th row has i+1 elements.
- Each row starts and ends with 1.
- We can consider the Pascal Triangle to be a 2D array, where pascal[i][j] gives us the value of the element in i-th row and j-th column of the Pascal Triangle.
- Except for the first and last column of each row, all other columns can be calculated as pascal[i][j] = pascal[i-1][j] + pascal[i-1][j-1].

For example, here are the first 10 rows of Pascal Triangle.

```
                  1
                1   1
              1   2   1
            1   3   3   1
          1   4   6   4   1
        1   5  10  10   5   1
      1   6  15  20  15   6   1
    1   7  21  35  35  21   7   1
  1   8  28  56  70  56  28   8   1
1   9  36  84 126 126  84  36   9   1
```

Ok. Now that we have revised what a Pascal Triangle is, let us talk about a new kind Pascal Triangle, called “Lucky Pascal Triangle”. Legend says that a Lucky Pascal Triangle is a Pascal Triangle containing only elements that are divisible by the ultimate lucky number 7. All other elements that are not divisible by 7 are converted to 0. So, the first 10 rows of a Lucky Pascal Triangle will be:

```
                  0
                0   0
              0   0   0
            0   0   0   0
          0   0   0   0   0
        0   0   0   0   0   0
      0   0   0   0   0   0   0
    0   7  21  35  35  21   7   0
  0   0  28  56  70  56  28   0   0
0   0   0  84 126 126  84   0   0   0
```

Fascinating right? It seems like there is some kind of pattern in the Lucky Pascal Triangle. We need to investigate more.

As the first step of the investigation, we need to know how many non-zero elements there are in a Lucky Pascal Triangle with N rows. Can you please help us?

Since the answer can be huge, please output your result modulo 10^9+7.

Input

The first line of the input contains a single integer T (1 <= T <= 10^5) denoting the number of test cases. Next T lines follow with a single non-negative integer N (N <= 10^18).

Output

For each value of N, print the test case number and output the number of non-zero elements in the Lucky Pascal Triangle with N rows, modulo 10^9+7. See sample input/output for more details.

Sample Input/Output

Input	Output
5 1 5 10 15 100000	Case 1: 0 Case 2: 0 Case 3: 18 Case 4: 43 Case 5: 653881477

The Ministry of Communication has an extremely wonderful message system, designed by the President himself. For maximum efficiency, each office in the Ministry can send a message directly to some, but not necessarily all, other offices. These connections are not always two-way, because sometimes one office should have the power to send a message to another office, but the other office should not have the power to send one back. This may seem unfair, or even illogical, to uneducated people, but this is the will of the President. There are so many offices now that the situation has become rather confusing, so the President has decided to reorganize the Ministry to be better adapted to the message system.

The President will divide the Ministry into new departments based on two simple principles:

1. If A and B are in the same department then A can transmit a message to B, and B can transmit a message to A.
2. If A can transmit a message to B, and B can transmit a message to A, then A and B are in the same department.

How many departments will the reorganized Ministry have?

Input

Input starts with a line containing a single integer N, with $0 < N \leq 100$. This tells you how many test cases there will be.

Each following pair lines contains a single test case. The first line of each test case contains an integer n, with $1 < n \leq 200$. This is the number of offices the Ministry has. The second line starts with an integer e with $0 < e < \frac{n^2}{4}$. This tells you how many individual direct (and directed) connections the Ministry has. Following this are e pairs of integers a, b, with $0 \leq a < b \leq n - 1$. These pairs indicate that there is a connection from a to b. There is at most one direct connection going out from one office and into another.

Output

Each line of output is an integer saying how many departments the Ministry corresponding to that test case will have.

Sample Input/Output

Input	Output
3 6 2 0 5 5 0 5 7 0 1 0 2 1 0 1 3 2 4 3 1 4 2 3 4 0 1 0 2 1 0 1 2	5 2 2

Crassus, the richest man in the world, invested some of his money with the Very Legitimate International Bank. The Bank offered a remarkable interest rate. They promised that given an initial investment of x, the balance at the beginning of the nth year would be x^n (counting the first year as year 1).

At the beginning of the 3rd year, there is a problem. It turns out that this investment opportunity is too good to be true, and it is actually a fraud. The Bank has spent all the money, and the directors have disappeared. Since Crassus is very rich, the Government decides to help him. They will pay him back his initial deposit out of taxpayers' money.

The Bank has lost all records of Crassus' original deposit, but does have information about what Crassus' current deposit value should be. This information is stored on 3 separate computers. Unfortunately, each computer only has a limited amount of memory, and is also very badly designed, so each computer stores integers modulo N_i, for $i = 1,2,3$. Though these values are all large enough to correctly store the initial value x, Crassus now has so much money 'invested' with the Bank that the computers don't have enough memory to store it correctly. I.e. $x^3 > N_i$ for all $i = 1,2,3$.

As the government official in charge of giving Crassus his initial deposit back, you must find the value of the original x that he invested. You know the numbers N_1, N_2, N_3 and the value $x^3 \, mod \, N_i$ for all i. You also read in the documentation for the computers that the numbers N_1, N_2, N_3 have the property that if p is a prime number and p divides N_i, then p does not divide N_j for all $i \neq j$.

Input

The first line contains a single number T indicating the number of test cases $(0 < T \leq 10)$.

The next $2T$ lines of input come in pairs as follows. The first line of each pair contains three numbers N_1, N_2, N_3 separated by spaces $(0 < N_i < 2^{21}$ for all $i)$.

The second line of each pair contains the values $x^3 \, mod \, N_i$ for each i, again separated by spaces.

Output

The value of x for each test case, written out in full, each on a new line.

Sample Input/Output

Input	Output
2 6 11 19 5 4 11 25 36 7 16 0 6	5 6

	Problem I Bowabowaukulipukuli Time Limit 1 sec	2018 ICPC Asia Nakhon Pathom Regional Contest

The Bowabowaukulipukuli is a microscopic organism (a microbe) that exists in a colony of cells. We will imagine the cells as unit squares. Initially there is only one cell. Then more cells come one by one. On arrival each new cell will place itself adjacent to one or more of the previous cells, and thus the colony of cells grows.

The microbes share nutrition among themselves. To send a nutrient from a cell A to a cell B, cell A sends the nutrient to one of its adjacent (horizontally or vertically) cells C, then C sends to one of its adjacent cells and so on until it arrives at B. The length of the path the nutrient travels is the number of cells that were involved in this process. The cells always send nutrients by the shortest path. If after adding a new cell, the shortest distance between some pair of previous cells changes, the colony becomes unstable. For example, consider following colony of 7 cells. The cell 1 came first, then cell 2, then cell 3 and so on.

		6
1	7	5
2	3	4

While it had 6 cells or fewer, the colony was stable. However, on the arrival of the 7th cell the colony becomes unstable. This is because before its arrival the distance between cell 1 and cell 5 was 5 (because all the cells numbered 1 to 5 are in the shortest path), and after adding the 7th cell the distance between these two cells becomes 3. *The Bowabowaukulipukuli is very intelligent microbe, so they will never make their colony unstable.* Thus the 'colony' shown above is not actually a valid colony.

A **super subcolony** is a subset of the cells in a colony which forms a rectangle (axis parallel). For example, in the above picture, cells 1-2-7-3 form a super subcolony. And if you count carefully you will find 21 super subcolonies in the picture.

You are given a Bowabowaukulipukuli colony which has a rectangular shape with R rows and C columns. The arrival order of each of the cells is given (from 1 to RC, here RC stands for R multiplied by C). After the arrival of each cell you are to output the number of super subcolonies in the colony.

Input:

The first line of the input contains T, the number of test cases (1 <= T <= 10). Hence follows T test cases.

Each test case starts with R and C, the number of rows and number of columns respectively (1 <= R, C <= 1000). Then follows R lines with C integers each. The numbers are unique and they range from 1 to RC, denoting the arrival order of the cells, and their positions in the final colony.

Output:

For each test case, output first the case number, then following this a number calculated using the formula

$$\sum_{i=1}^{RC} (ans[i] * 184903^{i}) \bmod 3428977,$$

where $ans[i]$ is the number of super subcolonies there are after the i^{th}cell has been placed.

For your convenience we will show here the results of the intermediate calculation that finds the values $ans[i]$ for all i, though this is not part of the final output.

Sample Input/Output

Input	Output
2 2 2 1 2 3 4 2 3 1 2 3 4 5 6	Case 1: 433509 Case 2: 3004506

Explanation:

The input, described above, is a situation with 2 test cases, where the first test case involves a 2 by 2 grid, and the grid is filled by cells in this order (1, 1), (1, 2), (2, 1), (2, 2). Their intermediate calculations that find the values $ans[i]$ for all i are as follows.

Intermediate Calculation
Case 1:
1 3 5 9
Case 2:
1 3 6 8 12 18

Problem J

Floating-Point Hazard

Time Limit 1 sec

2018 ICPC
Asia Nakhon Pathom
Regional Contest

Given the value of *low, high* you will have to find the value of the following expression:

$$\sum_{i=low}^{high} \left(\sqrt[3]{(i+10^{-15})} - \sqrt[3]{i}\right)$$

If you try to find the value of the above expression in a straightforward way, the answer may be incorrect due to precision error.

Input:

The input file contains at most **2000** lines of inputs. Each line contains two integers which denote the value of **low**, **high** (**1 ≤ low ≤ high ≤ 2000000000** and **high-low ≤ 10000**). Input is terminated by a line containing two zeroes. This line should not be processed.

Output:

For each line of input produce one line of output. This line should contain the value of the expression above in exponential format. The mantissa part should have one digit before the decimal point and be rounded to five digits after the decimal point. To be more specific the output should be of the form d.dddddE-ddd, here d means a decimal digit and E means power of 10. Look at the output for sample input for details. Your output should follow the same pattern as shown below.

Sample Input/Output

Input	Output
1 100 **10000 20000** **0 0**	**3.83346E-015** **5.60041E-015**

Little Oregon loves to play in the forest, but tonight he got lost on his way back home. While trying to find his trail, he found a stream. No no, it was not a stream of hot water, rather it was a stream of 32 bit integers!

Getting super excited, he sat by the stream and kept observing as the numbers kept coming. To his surprise, after coming out the numbers kept disappearing after a while! He asked Paka, the biggest tree by the stream, why the numbers were disappearing, and Paka replied that each number of the stream comes with an expiration time. Beyond its expiration point, that number gets eaten by the forest monster immediately. Each number has its own expiration time, and Paka offered that, as each number comes out, he will let Oregon know the expiration time of the number instantly. In exchange, sometimes Oregon will have to calculate the K'th smallest number of the numbers that has not expired yet, where K will be given by Paka.

Oregon has recently signed up for ICPC Regional contest, so he thinks the deal Paka offered will be great for his practice. So let's help Oregon with his regional al preparation. :)

Input:

Input starts with an integer T, then T cases follow.

Each case starts with an integer, which represents the number of events E. Then E lines follow, each with one event.

Each event is a list of integers separated by spaces. A line of an event will be one of the following two types:

1. OP(= 1): CURRENT_TIMESTAMP VALUE END_TIMESTAMP
   This event starts with 1 which denotes the event type, followed by 3 integers:
   a. CURRENT_TIMESTAMP: the current timestamp
   b. VALUE: the value of the stream
   c. END_TIMESTAMP: the timestamp where this value will expire.
2. OP(= 2): CURRENT_TIMESTAMP K
   This event starts with 2 which denotes the event type, followed by 2 integers:
   a. CURRENT_TIMESTAMP: the current timestamp
   b. K: the position for the value that Paka is asking for.

Explanation:
Time is represented as a sequence of timestamps, which are 32 bit integers, the greater the timestamp implies the later the time. CURRENT_TIMESTAMP is the timestamp of a number coming out of the stream, or the timestamp of a query given by Paka. In the given inputs, the CURRENT_TIMESTAMPS will be strictly increasing between consecutive events.

And a number cannot expire before coming out of the stream, hence END_TIMESTAMP >= CURRENT_TIMESTAMP.

Constraints:

1. 1 <= T <= 100
2. 1 <= E <= 10^5
3. 1 <= CURRENT_TIMESTAMP <= END_TIMESTAMP <= 10^9
4. 1 <= VALUE <= 10^6
5. 1 <= K <= 10^9

Output:

For each case, print the case number first. Followed by one line per event of type 2 for the case. For each event of type 2, print one integer, the K'th smallest number currently in the stream. Events are in the stream at time t if they entered the stream at time <= t, and their END_TIMESTAMP value is >= t. If no such number exists, then print -1.

Sample Input/Output

Input	Output
1 10 1 1 10 7 1 2 9 4 2 4 1 1 5 2 6 2 6 2 2 7 2 1 8 2 20 1 9 1 15 1 10 3 13 2 11 3	Case 1: 9 10 -1 3

	Problem L Largest Allowed Area Time Limit 1 sec	2018 ICPC Asia Nakhon Pathom Regional Contest

A company is looking for land to build its headquarters. It has a lot of money and can buy as many land patches as it needs. Its goal, however, is finding the largest square region containing no forest. Unfortunately, there is no such region that is large enough for the headquarters they want to build.

After negotiation with the government and the evaluation of environmental impacts, the government allows the company to purchase land with at most one forest patch. In other words, the company's goal is now finding the largest square region containing at most one forest patch.

To facilitate the search process, the company creates a map in the form of a 2D table consisting R rows and C columns. In this 2D table, each entry represents a land of patch where 0 corresponds to a non-forest patch and 1 to a forest patch. Unfortunately, the map may have up to 1,000 x 1,000 entries and it is not a good idea to manually look for the largest allowed square region. This is where your skill comes into play. Write an efficient algorithm to find such a square region.

Input:

The first line is a positive integer T <= 20 representing the number of test cases. For each case, the input is formatted as follows.

First line	R C where R and C represents the number of rows and columns in the map. Also, 5 <= R, C <= 1,000
Next R lines	Each line represents a row in the map from the first to last. It has C numbers which are 0 or 1, separated by one space.

Note: there is at least one non-forest patch in each test case.

Output:

There are T lines in the output. Each line is the number of rows in the largest allowed square region for each case.

Sample Input/Output

Input	Output
2 10 20 0 0 0 0 0 **0** 0 0 0 0 0 0 0 **0** 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 **0** 0 0 0 0 0 0 0 **0** 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 10 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 **0** 0 0 0 0 0 **0** 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 **0** 0 0 0 0 0 **0** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	9 7

Hint: the highlighted numbers in the example indicate the corner of a largest allowed region. There may be multiple such regions.

Problem Setters

Pinyo Taeprasartsit, Dept. of Computing, Faculty of Science, Silpakorn University
Thanawin Rakthanmanon, Dept. of Computer Engineering, Faculty of Engineering, Kasetsart University
Robert Egrot, Faculty of ICT, Mahidol University
Gun Srijuntongsiri, School of ICT, SIIT
Nattee Niparnan, Dept. of Computer Engineering, Faculty of Engineering, Chulalongkorn University
Shahriar Manzoor, Dept of CSE, Southeast University
Derek Kisman, Laserlike
Monirul Hasan, Dept of CSE, Southeast University
Md Mahbubul Hasan, Hidden Affiliation
Hasnain Heickal, Dept. of CSE, University of Dhaka
Shafaet Ashraf, Traveloka.com
Sabit Zahin, Newscred
Sadia Nahreen, Hidden Affiliation
Mohammad Nafis Sadique, Grab Singapore
Mohammad Samiul Islam, Hidden Affiliation
Mohammad Ashraful Islam, Mawlana Bhashani Science and Technology University
Raihat Zaman Neloy, NewsCred,