Problem A. Stara Zagora

Stara Zagora is known as the city of linden, poets, and straight streets. Under the scent of the linden trees, young men recite poems to their beloved ones. Elly is also among them (the beloved ones, not the lindens – she cannot be a tree since she has a cycle). In contrast to the other girls, though, she's more fascinated by the straight streets, than the poems of the young poets.

"How efficient can everything be!" – though Elly. If every street is one-way, the traffic would be immensely revealed. Of course, to make the infrastructure good, there should be a quick path between pairs of important objects. Examples of such objects would be the rail station and the central hotel; the zoo and the vet university; the stadium and the hospital; the high-school and the brewery.

The girl now plans to propose to the municipality to make all the streets one-way, in such a way, that for all pairs of important objects (A_i, B_i) there is a path from A_i to B_i that requires at most one turn (which means it uses at most two streets – one horizontal and one vertical).

For simplicity we'll represent the city as a rectangle with N horizontal streets (directed west-east or east-west) and M vertical streets (north-south or south-north). We'll number the horizontal streets with the integers 1 to N and the vertical ones – with the integers 1 to M. There are K (ordered) pairs of important objects (A_i, B_i), for which the mayor insists on having a quick path (using at most two streets) between the first and the latter.

Your task is to determine if all streets can be made one-way in such a way, that all K requirements are satisfied.

Input

On the first line of the standard input will be given one integer T – the number of tests that your program must process. Each test starts with a line, containing three integers N, M, and K – the number of horizontal streets, the number of vertical streets, and the number of important pairs of objects, respectively. Each of the next K lines contains four integers AR_i, AC_i, BR_i, and BC_i – the horizontal and vertical street of the origin and the horizontal and vertical street of the destination object of each of the requirements. It is guaranteed, that the origin and destination do not coincide ($AR_i \neq BR_i$ or $AC_i \neq BC_i$).

Output

For each test print on a line on the standard input one string – "Yes", if the task can be completed, or "No" otherwise.

Constraints

$1 \leq T \leq 10$;
$1 \leq N, M \leq 50$;
$1 \leq K \leq 200$;
$1 \leq AR_i, BR_i \leq N$;
$1 \leq AC_i, BC_i \leq M$.

Example

Input	Output
3 9 8 6 3 3 5 5 5 6 4 3 4 5 3 3 4 3 5 5 5 6 3 3 3 2 4 5 7 7 2 2 2 7 7 7 7 2 2 8 11 4 2 2 2 7 7 2 7 7 7 7 2 2 5 4 6 2	No Yes No

Problem B. Cake

Mimmy's favourite teacher cel1o has a jubilee! For this occasion she decided to make a cake for him. Mimmy rolled her sleeves up, found the most bestest recipe and started preparing it. All was going according to flan until she reached the final row. There the recipe said. "Add *X* milliliters of milk". Uh-oh. As it turns out Mimmy had no measuring cup. How is she supposed to measure *X* milliliters of milk then? Luckily she kept calm and remembered she has a bottle, that she knows can fit *A* milliliters of milk, and with the sink that she knows can fit *B* milliliters of milk.

And so she invented the following algorithm – she will fill the bottle to the top and then she will pour it all in the sink. She will repeat this operations until the sink gets full (at this moment probably there will be some milk left in the bottle). After that she will remove the stopper and she will wait until the milk is gone in the tubing, she will place the stopper again and she will pour the milk left in the bottle in the sink. She will continue doing this until the quantity left in the bottle is such that she can measure exactly *X* milliliters of milk. Note that it is possible that *X* is more than the capacity of the bottle. Then Mimmy will pour some number of full bottles in the cake and she will need to measure the remainder. Of course Mimmy will never use the milk she's poured in the sink (eeeewww).

As Mimmy loves milk she has a virtually inexhaustible reserve. Still Mimmy is GREEN so she wants to least the minimal amount of milk possible.

Mimmy decided to write a program which tells her how many times she must fill the bottle and empty the sink so that she reaches the required amount of *X* milliliters of milk. Unfortunately she is covered in flour, eggs, beef, tomato souse, cream and chocolate so she can't code right now. Thus she asks you to write the program in her stead.

Input

On the first line of the standard input there will be a single integer *T* – the number of tests that you need to solve. *T* lines follow each of them describing one test. Each line will contain 3 integer numbers – *A*, *B* and *X* – the bottle capacity, the sink capacity and the required milk amount respectively. The numbers will be separated by a single space.

Output

For each test on a separate line output two numbers *NB* and *NS* – the number of times Mimmy will need to fill the bottle and the number of times she will need to empty the sink respectively. If Mimmy can't measure *X* milliliters no matter what she does print the string "IMPOSSIBLE" (without quotes).

Constraints

$1 \leq A, B, X \leq 1\,000\,000$;
$1 \leq T \leq 100\,000$.

Example

Input	Output
3 5 7 2 6 9 4 100 999 360	6 4 IMPOSSIBLE 603 60

Explanation of the first test

Mimmy fills the bottle ones and pours it in the sink, after that she fills the bottle second time, but the sink fits just 2 milliliters of it. Thus there are 3 milliliters left in the bottle. Then Mimmy empties the sink and places the stopper back. Now Mimmy purs the remaining 3 milliliters and fills the bottle again. This time the sink fits 4 milliliters from the bottle(and 1 is left in the bottle). Mimmy empties the sink ones more and then pours 1 milliliter in it. She fills the bottle again and pours it all in the sink. After that she fills the bottle ones more but pours only 1 milliliter in the sink. After she empties the sink again she pours the 4 millliliters left in the bottle. Now Mimmy fills the bottle ones more and pours 3 milliliters from it in the sink. After performing this action there are exactly 2 milliliters left in the bottle and Mimmy can continue cooking.

Problem C. D2

Let us denote with $f(w)$ the sum of the positive integers x, for which x^2 divides w. For example $f(36) = 1 + 2 + 3 + 6 = 12$. Write a program which calculates $f(w)$.

Input

On the first line a single integer q is given – the number of tests. q lines follow each specifying a single number w_i .

Output

Output q lines, where the i-th of those lines contains $f(w_i)$.

Constraints

$1 \leq w \leq 10^{18}$
$1 \leq q \leq 300$

Example

Input	Output
2 36 4608	12 124

Problem D. Employees of the month

Fiki and Kiro work at a warehouse. Their job is to carry boxes from one container to another.
There are *N* containers situated in a line and each one is 1 meter apart from its adjacent containers. The boss gives orders to Fiki and Kiro to transport a box from one container to another in the format "N_i N_j", where *i* and *j* are the positions of the start and destination containers, respectively. When either worker carries a box to the destination container, he must remain there until the next order.
The problem is that both of them are too lazy and want the job to be done with minimum effort. Write a program that finds the minimum distance (in meters) they have to walk to execute their boss' orders.
Every order can be executed by only one worker.
The initial travel distance of both workers to their starting positions is 0.

Input

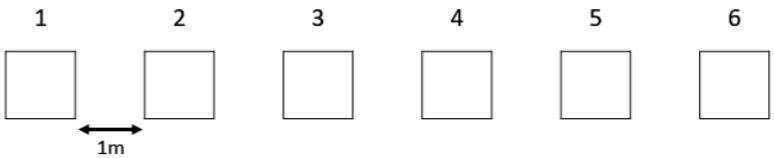
On the first input line in the file the number of test cases *T* will be given. Each test case contains on the first line two space-separated integers – *N* (number of containers) and *M* (number of orders). The next *M* lines hold the orders of the boss, each in the format "*Ni Nj*", describing that a box form container *i* should be moved to container *j*.

Output

For each test case print the answer - **minimum distance** the workers have to walk on a single line.

Example

Input	Output
1 6 4 2 6 4 3 5 2 3 5	11



Explanation:

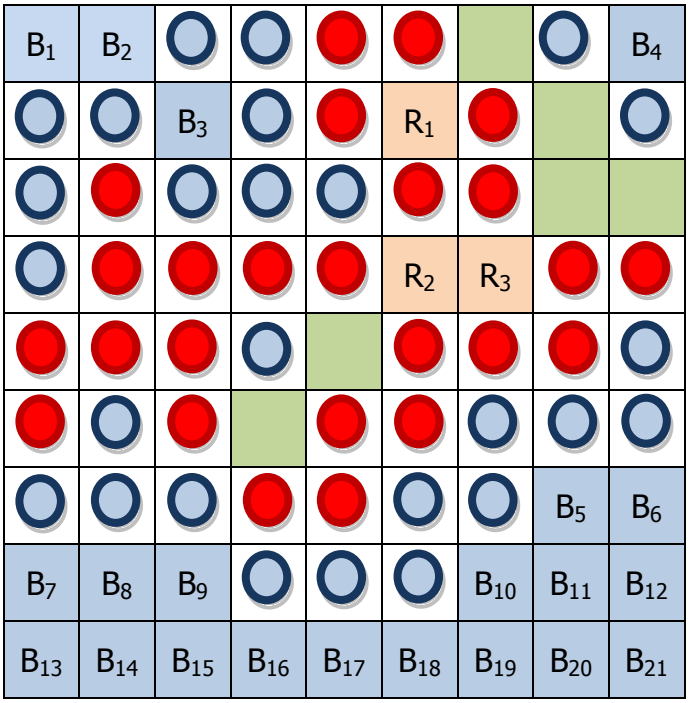
1. Worker 1 (W1) starts at 2 and moves box from 2 to 6: **4 meters**.
2. Worker 2 (W2) starts at 4 and moves box from 4 to 3: **1 meter**.
3. W1 walks 6 -> 5 and moves 5 -> 2: 1 + 3 = **4 meters**.
4. W2 is already at 3 and moves 3 -> 5 = **2 meters**.

The total distance walked by both is **11 meters.**

Problem E. Fence

In the game Fence, two players one after another place red and blue pieces on a board with *N*x*N* size and entirely white fields. Each one tries to enclose as much as possible territory (i.e. regions of unfilled fields). At the end of the game, the score of each player is the total enclosed area with his pieces. Write a program that calculates the outcome of each player and determines the winner, if the location of the red and blue pieces on the board is known at the end of the game. The program should be able to process multiple test cases.

For example, the Figure shows a 9x9 game board and the distribution of the red and blue pieces. With R_1, R_2, R_3, the empty fields are marked, that the player with the red pieces has managed to surround. With B_1, B_2,…,B_{21}, the empty fields are marked, that the player with the blue chips has managed to surround. Unoccupied fields are left free. The blue pieces have enclosed 21 fields, and the red pieces - 3 fields.



Two fields on the board with coordinates (x_1, y_1) and (x_2, y_2) are adjacent if $| x_1-x_2 | + | y_1-y_2 | = 1$
A connected region of unfilled fields belongs to the territory of one of the players if all adjacent filled fields belong to that player (see Figure). The score of each player is formed by the total number of empty fields that he has managed to surround.

Input

The number of test cases is entered from the first line of the input. For each test case the first line contains three integers *N*, *B* and *R*, separated by a space. The number *N* is the size of the board, *B* is the number of blue pieces placed on the board, *R* is the number of red pieces placed on the board. The second line contains *B* pairs of integers $x_1\ y_1\ \ldots\ x_B\ y_B$ (where $1 \le x_i, y_i \le N$) that indicate the location of the blue pieces on the board. The third line contains *R* pairs of integers $x_1\ y_1\ \ldots\ x_R\ y_R$ (where $1 \le x_i, y_i \le N$) that indicate the location of the red pieces on the board. It is guaranteed that no more than one piece is allowed to occupy a playing field.

Output
On a single line of the output file, for each test case, print string: " *Bs*:*Rq* ", where *s* is an integer representing the number of fields enclosed by the blue pieces, and q is an integer representing the number of fields enclosed of the red pieces.

Constraints
$1 \leq N \leq 19$
$B \geq 0, R \geq 0$ и $1 \leq B + R \leq N^2$

Example

Input	Output
2 9 26 25 1 3 1 4 1 8 2 1 2 2 2 4 2 9 3 1 3 3 3 4 3 5 4 1 5 4 5 9 6 2 6 7 6 8 6 9 7 1 7 2 7 3 7 6 7 7 8 4 8 5 8 6 1 5 1 6 2 5 2 7 3 2 3 6 3 7 4 2 4 3 4 4 4 5 4 8 4 9 5 1 5 2 5 3 5 6 5 7 5 8 6 1 6 3 6 5 6 6 7 4 7 5 5 12 4 1 1 1 2 1 3 2 1 2 3 3 1 3 3 4 1 4 3 5 1 5 2 5 3 1 4 2 4 3 4 3 5	B21:R3 B3:R2

Problem F. Skyscraper

An international consortium of construction companies plans to build the tallest skyscraper in the world. The preliminary plans are for the building to be declared a new part of the campus to the Angel Kunchev University, and the completion of the building is expected to be no earlier than 2020. One of the construction companies offers the skyscraper to be not only the tallest in the world, but the building with the most floors. A major competitor in such a race would be the Jeddah Tower, which will also be completed in 2020. The Jeddah Tower is expected to be at least 170 floors, and in order for the skyscraper in Rousse to be named as the sole leader in this ranking, it must exceed that number. For this reason, the consortium of construction companies assigned to the "Angel Kanchev" University the task of preparing a preliminary plan for the building. Under the preliminary arrangements, the building will be a steel, concrete and glass construction (in the form of a rectangular parallelepiped). The construction will consist of 2 types of monolithic steel gratings (rectangular parallelepipeds measuring 5m x 5m x 5m and 10m x 5m x 5m). The base of the building will be 10m x 5m and each floor will be 5m high. The number of floors will not be fixed at the start of construction and will be clear once the construction of the main competitor is completed. The preliminary construction assignment should describe the number of combinations of possible arrangements of monolithic steel grids that will build the skyscraper.
Write a program that calculates the number of combinations of monolithic grids on which to build a skyscraper with a certain number of floors.
The program should handle several test cases.

Input
The first line in the input file contains an integer t – the number of test cases.
At each successive line is the number of floors - f, needed to calculate the combination of monolithic grids.

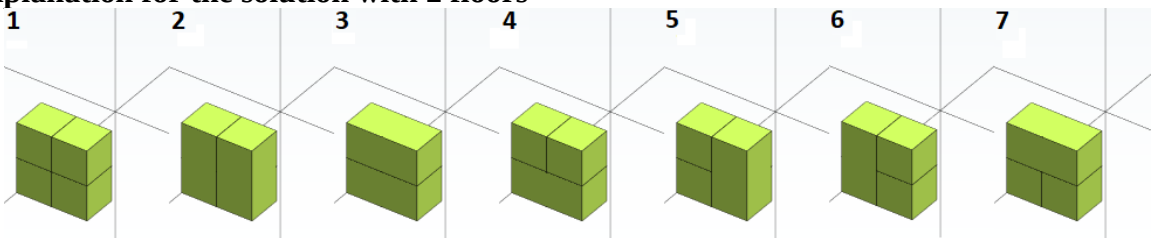
Output
For each test case, a line containing the calculated answer is displayed in the output file.

Constraints
$1 \leq t \leq 100$; $1 \leq f \leq 200$

Example

Input	Output
4 2 1 4 3	7 2 71 22

Explanation for the solution with 2 floors



Problem G. X2017

A string containing decimal digits is given. Write a program, which computes in how many ways it is possible to form 2017 from four of given digits arranging them in the same sequence as in initial string.

Input
The first line of the standard input contains the number of test cases. Each test case consists of one line containing a string of decimal digits.

Output
For each test case output the computed count on a different line of the standard output.

Constraints
The length of the given strings is at most 4000.

Example

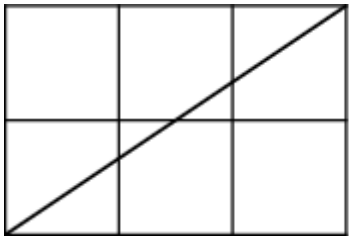
Input	Output
2 352906120127 22001177	4 16

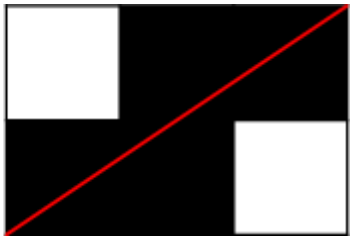
Explanation of the first test case

```
352906120127
..2.0.1....7
..2.0....1.7
..2.....01.7
.......201.7
```

Problem H.Raster

Pesho is a computer graphics student. He has a homework which is very easy for him. He has to make a program that draws a line from point (0, 0) to (*a*, *b*), where integers *a* and *b* are the input of the program.
He uses the following algorithm. He divides the plane into squares 1x1 – these squares are pixels. When the line from (0, 0) to (*a*, *b*) intersects a square in more than one point, the square (pixel) will be black. Otherwise it will be white. Look at the example:





Pesho did his homework in 30 minutes and now he is interested in a slightly different problem. Given an integer *N*, for how many different inputs does his algorithm produce exactly *N* black pixels?
More precisely, he is only interested in lines beginning in (0, 0) and ending in (*a*, *b*), where both a and b are positive integers. Given *N*, find out how many of these lines will produce exactly *N* black pixels.

Input
The first line of the input file contains an integer *T* specifying the number of test cases. The following *T* lines contain a single positive integer *N*.

Output
For each test case output one line with one integer – the number of lines that use exactly *N* black pixels.

Constraints
$a > 0, b > 0$
$1 \leq N \leq 10^{14}$

Example

Input	Output
2 2 6	3 11

Explanation for the first test case
The three good lines are those ending in (1,2), (2,1), and in (2,2).

Задача I. Divisor

We say that an integer *A* is better than another integer *B* if the sum of the digits of *A* is greater than the sum of the digits of *B*. If the sums of the digits of the two numbers are equal, the better integer is that, which is less than the other. Naturally, if *A* and *B* match, we can not determine which number is better.

Write a program that finds the worst divisor of a given integer *N*. It is assumed that the 1 and the number *N* itself are also divisors of *N*.

Input

On the first line the number of test cases is given. Each of the following lines contains an integer *N*.

Output

For each test case output a line with a single integer – the worst divisor of *N*.

Constraints

$0 < N \leq 10^{50}$

Example

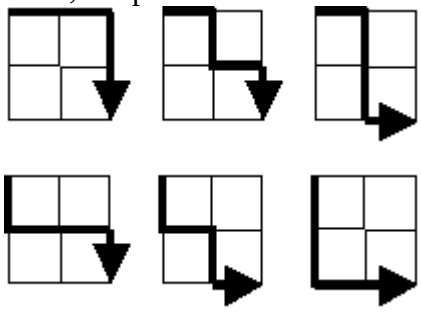
Input	Output
2 2325 75810	1 10

Problem J. Paths

A square grid with dimensions *N*x*N* is given. Starting in the top left corner, and only being able to move to the right and down, there is finite number of paths to the bottom right corner.
For instance, when *N*=1, the possible solutions are 2:



When *N*=2, the possible solutions are 6:



Write a program that calculates the exact number of the possible paths for a given integer *N*.

Input
The first line contains an integer *t* – the number of test cases. Each of the following *t* lines contains a single integer N, representing the size of the grid.

Output
For each test case output a line with exactly one positive integer – the number of the possible paths to reach the bottom right corner from the top left one.

Constraints
$0 \leq N \leq 30$

Example

Input	Output
1 2	6

Problem K. Connected, not connected

N villages of a region labeled from 1 to *N* are connected by roads. After heavy snowing only *M* of the roads, each of them connecting 2 different villages, are passable. Nervous citizens which have to travel in the region continuously call 112 with questions of the kind „Could we pass from *X* to *Y*?“ In the same time people of the road maintenance succeed to clean some of the roads connecting directly two villages and report for this on 112 too: „Road from *U* to *V* is now passable.“ From other side, citizens that trust 112 information and go to travel reach a road which is covered by huge amount of snow again and call angry 112: „The road from *U* to *V* is impassable.“ Write a program to help operators of 112 to answer quickly questions of citizens for actual state of the roads network.

Input

On the first line of the input the number of the test cases will be given. Each test case starts with line containing the numbers *N* and *M*. Each of the next *M* lines contains the labels of two villages connected with a passable road. A line with the number *Q* of the calls of citizens and the road maintenance follows as well as *Q* lines with the content of the calls – code of call and two concerned villages. The code of citizens' questions is 1, the code of roads maintenance people call is 2, and the code of angry citizens informing for impassable road is 3.

Output

For each test case the program has to print on the standard output a bit string with as many characters as the number of calls of kind 1. The sign 0 means that the answer of the current question is „Not“, and the answer 1 means that the answer is „Yes“.

Constraints

$2 \leq N \leq 1000$; $1 \leq Q \leq 1000000$.

Example

Input	Output
1 9 8 1 2 3 4 5 6 7 8 9 5 7 2 8 2 6 9 8 1 1 8 1 6 2 2 7 1 1 4 7 2 2 3 1 4 7 3 3 4 1 4 7	10010

Problem L.Repair

The road network in Ruse consists of crossroads and two-way streets. Each street connects exactly two crossroads, and no two streets connect two identical crossroads. The local authorities plan to repair the most of the streets. When repairing a street, it closes for traffic. The streets are not being repaired at the same time, but the requirement is if there was a road between two crossroads, it would still exist road after the closure of a street.
Write a program that finds these streets, which should not be repaired.

Input
On the first line one integer will be given – the number of test cases.
Each test case starts with a line, containing two integers *N*, *M* – the number of crossroads and the number of streets. Each of the next *M* lines contains two integers – the numbers of crossroads which the street connects.

Output
For each test case output on separate lines information about detected streets. The numbers of the crossroads describing each street should be displayed in ascending order. The information about streets should be displayed in ascending order of the first crossroad. If the first numbers are equal – in ascending order of the second crossroad.

Constraints
$1 \leq N \leq 2000$
The crossroads are numbered from 1.

Example

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