GRAN PREMIO FASE I

Problem A. Toby the adventurer

Source file name:	adventurer.c, adventurer.cpp, adventurer.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Manuel Felipe Pineda - UTP Colombia`

Toby is a great adventurer. Today he is trying to explore "Bitland" (a new country that will be remembered after Toby's exploration).

Bitland is divided into N small cities and M unidirectional roads between cities.

Toby begins the adventure at the city R, and after that he goes to any city R', if this new city (R') is not known by Toby, a road between R and R' is needed and he must pay a cost (in terms of adventure power) associated to the road. Otherwise, if Toby wants to go to a known city he does not need pay anything, even if there is no road from the current city to the target city (like teleportation)... is not Toby so cool?

Toby keeps traveling between cities until he reaches every city in Bitland. After this moment Toby goes to home, happy and eager for new adventures.

Wait! Where is the problem?

Did you remember that Toby has to pay for each road that is used to disclose a new city? Help Toby to minimize this cost (the sum of all power paid), because he needs as much energy as possible for his new adventures.

Input

The input starts with an integer $1 < T \leq 100$ indicating the number of test cases.

Each test case begins with three integers $3 < N \leq 10\,000$, $3 < M \leq N$, $0 \leq R < N$ denoting the number of cities, number of roads and initial city, respectively. Followed by M lines which contain three integers, $0 \leq u$, $v < N$, $1 \leq w \leq 10\,000$. These numbers denote a road from the city u to the city v with cost w.

Note that there could be several roads between the same pair of cities

Output

Print one line with the total cost for the adventure, followed by $N - 1$ lines with the chosen roads in the same format that was given in the input:

u v w - three space separated integers denoting a road from u to v with cost w.

If there are several answers, print any of them.

If there is no way to visit all the N cities, print "impossible" without quotes.

Example

Input

```
3

5 5 0
0 1 1
0 2 100
1 3 2
3 2 3
2 4 4

5 5 4
0 1 1
0 2 100
1 3 2
3 2 3
2 4 4

4 4 0
0 1 3
0 2 4
3 1 1
2 3 1
```

Output

```
10
0 1 1
3 2 3
1 3 2
2 4 4
impossible
6
3 1 1
0 2 4
2 3 1
```

Use faster I/O methods

Problem B. The book thief

Source file name:	book.c, book.cpp, book.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Hugo Humberto Morales Peña - UTP Colombia`

On February 18, 2014, Red Matemática proposed the following mathematical challenge on their twitter account (@redmatematicant): "While Anita read: *The book thief* by Markus Zusak, She added all the page numbers starting from 1. When she finished the book, she got a sum equal to 9.000 but she realized that one page number was forgotten in the process. What is such number? and, how many pages does the book have?"

Using this interesting puzzle as our starting point, the problem you are asked to solve now is: Given a positive integer s ($1 \leq s \leq 10^8$) representing the result obtained by Anita, find out the number of the forgotten page and the total number of pages in the book.

Input

The input may contain several test cases. Each test case is presented on a single line, and contains one positive integer s. The input ends with a test case in which s is zero, and this case must not be processed.

Output

For each test case, your program must print two positive integers, separated by a space, denoting the number of the forgotten page and the total number pages in the book. Each valid test case must generate just one output line.

Example

Input	Output

`1`
`2`
`3`
`4`
`5`
`6`
`9000`
`499977`
`49999775`
`0`
	`2 2`
	`1 2`
	`3 3`
	`2 3`
	`1 3`
	`4 4`
	`45 134`
	`523 1000`
	`5225 10000`

Use faster I/O methods

Problem C. Numeric Center

Source file name:	center.c, center.cpp, center.java
Input:	Standard
Output:	Standard
Author(s):	Hugo Morales, Sebastián Gómez & Santiago Gutierrez - UTP Colombia

A numeric center is a number that separates in a consecutive and positive integer number list (starting at one) in two groups of consecutive and positive integer numbers, in which their sum is the same. The first numeric center is number 6, which takes the list $\{1, 2, 3, 4, 5, 6, 7, 8\}$ and produces two lists of consecutive and positive integer numbers in which their sum (in this case 15) is the same. Those lists are: $\{1, 2, 3, 4, 5\}$ and $\{7, 8\}$. The second numeric center is 35, that takes the list $\{1, 2, 3, 4, \ldots, 49\}$ and produces the following two lists: $\{1, 2, 3, 4, \ldots, 34\}$ and $\{36, 37, 38, 39, \ldots, 49\}$, the sum of each list is equal to 395.

The task consists in writing a program that calculates the total of numeric centers between 1 and n.

Input

The input consists of several test cases. There is only one line for each test case. This line contains a positive integer number n ($1 \leq n \leq 10^{14}$). The last test case is a value of n equal to zero, this test case should not be processed.

Output

For each test case you have to print in one line, the number of numeric centers between 1 and n.

Example

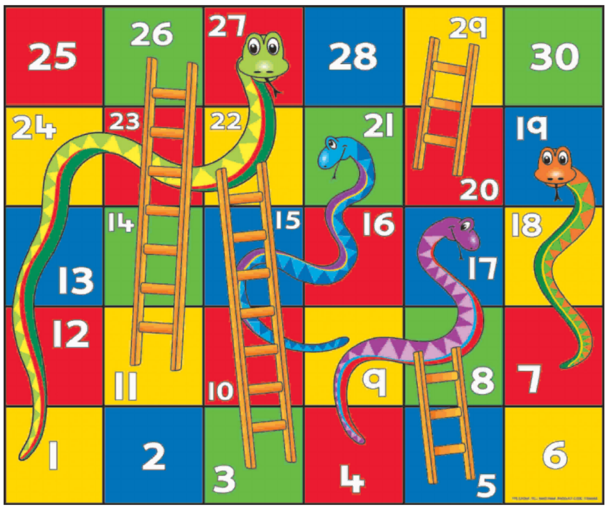
Input	Output
1	0
7	0
8	1
48	1
49	2
50	2
0	

Problem D. Snakes and Ladders

Source file name:	snakes.c, snakes.cpp, snakes.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Sebastián Gómez - UTP Colombia`

Snakes and ladders is a popular game for kids (and cute Dogs of course). Usually this game is played between multiple players but Toby does not like the other pups in his school, and wants to play alone. The game is very simple, Toby starts at position 1 of a board of height H and width W and the goal is to get to position $H \times W$.

Each turn Toby rolls a fair die and advances a number of positions equal to the result of the die. If at the end of a turn Toby lands at the bottom of a ladder he advances immediately to the top, and if Toby lands at the head of a snake then he goes back to the tail of the snake immediately as well.



Board of the third test case sample

Remember that a fair die is a die where the probability to get any outcome between 1 and 6 is the same. In the figure 1 you can see a sample board. To explain what happens when Toby is close to the finish let's make an example with this board. Let's suppose that Toby is at position 29. Then Toby rolls the die, if he gets one he advances to position 30 and wins. If he gets 2, he lands in 29 again (Advance one and go one back). If he gets, 3 he lands in 28 (Advance one and go two back). If he gets 4, he lands in 27 and then immediately goes to position 1 since he stepped in the head of a snake.

Now Toby wants to know how long will it take his game before it ends, and he asks you to compute the expected amount of turns (die rolls) before he wins. It is guaranteed that it is always possible to reach the goal of the board and that the maximum expected number of turns will not exceed 100 000. The starting cell will never be the base of a ladder and the target cell will never be the head of a snake.

Input

The input consists of several test cases. Each test case begins with a line with three integers W,H and S. Here W and H are as above and S is the number of snakes or ladders. Then follow S lines, each with two integers u_i and v_i meaning if you land in the cell u_i you have to go to cell v_i immediately. So if $u_i < v_i$ it is a ladder and if $u_i > v_i$ it is a snake. It is guaranteed that $u_i \neq u_j \forall i \neq j$ and $u_i \neq v_j \forall i, j$. Read input until end of file is reached, there will be a blank line after each test case.

- $1 \leq W, H \leq 12$
- $W \times H \geq 7$
- $0 \leq S \leq \frac{W \times H}{2}$
- $1 \leq u_i, v_i \leq W \times H$

Output

For each test case print in one line a single number consisting on the expected number of turns to finish the game. The answer will be considered correct if the difference with respect to the right answer is less than 10^{-2}.

Example

Input	Output
7 1 0	6.00000000
	13.04772792
6 5 0	19.83332560
6 5 8	
3 22	
17 4	
5 8	
19 7	
21 9	
11 26	
27 1	
20 29	

Problem E. Subset sum

Source file name:	subset.c, subset.cpp, subset.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Sebastián Gómez - UTP Colombia`

Given a set s of integers, your task is to determine how many different non-empty subsets sum up to a target value.

Input

The input consists of several test cases. The first line of each test case is a line containing two integers N and T, the number of items of the original set of integers and the target value. After that comes one line with the N integers s_i that belong to the original set s.

- $1 \leq N \leq 40$
- $-10^9 \leq T,\ s_i \leq 10^9$

Output

For each test case print on a single line an integer indicating the number of different non-empty subsets that sum up to the target value T.

Example

Input	Output

6 0
-1 2 -3 4 -5 6
5 0
-7 -3 -2 5 8
	4
	1

Explication

On the first test case the target is 0 and the following are the valid subsets: (2, 4, -1, -5), (2, 6, -5, -3), (4, -1, -3), (6, -5, -1). On the second test case the target is again 0, the only valid subset is: (-3, -2, 5)

Problem F. Toby and the strange function

Source file name:	strange.c, strange.cpp, strange.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Jhon Jimenez - UTP Colombia`

As is well known, Toby is a cute and smart dog, but this problem is too hard even for Toby. For this problem he needs to find a function f that receive two arguments, an integer n and a string S and return a string S', more formally $f(n, S) = S$'.

Input

The first line contains a single integer T denoting the number of test cases. Each case in the first line contains an integer n ($0 \leq n \leq 10^{18}$) and the second line contains S (the string only contains lowercase Latin letters), the length of S does not exceed 100 characters.

Output

For each test case you have to print in one line the string S', value of $f(n,\ S)$.

Example

Input	Output
3 1 abcd 2 abcd 4 abcd	dabc cdab abcd

Explication

$f(1,\ abcd) = dabc$
$f(2,\ abcd) = cdab$
$f(4,\ abcd) = abcd$

Can you help the poor dog in this complicated task?

Problem G. Grounded

Source file name:	grounded.c, grounded.cpp, grounded.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Sebastián Gómez - UTP Colombia`

Toby was behaving badly at little dog school and his teacher grounded him by asking him to solve a hard problem. Toby is given a number N, let's consider a set S of all binary strings of N bits. Let's also consider any subset P_i of S, let $XOR(P_i)$ be the XOR of all the elements of P_i. The XOR of the empty set is a binary string of N zeros.

As Toby is a very smart dog, and Toby's teacher wants Toby to spend a very long time working on the problem, he asks:

How many different subsets P_i of S exist such than $XOR(P_i)$ has exactly K ones?

Recall that the empty set and S itself are valid subsets of S.

Input

The input consist of several test cases. Each test case consists of a line containing the numbers N and K. The end of the test cases is given by the end of file (EOF).

- $1 \leq N \leq 10^6$
- $0 \leq K \leq N$

Output

For each test case print in one line the requested answer modulo $p = 10^9 + 7$.

Example

Input	Output
2 0 1 1	4 2

Explication

For the first test case the subsets of the strings of 2 bits with an XOR with zero ones is: {}, {00}, {01, 10, 11} and {00, 01, 10, 11}

For the second test case the subsets of the strings of 1 bit with an XOR with one is: {1}, {0, 1}

Problem H. Toby and the frog

Source file name:	frog.c, frog.cpp, frog.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Manuel Felipe Pineda - UTP Colombia`

Toby the dog is on the cell 0 of a numbered road, TJ the frog is on the cell number X of the same road. Toby wants to catch the frog, but he is not smart enough to count the distance from his current position to X, so he performs the following algorithm:

- Let *pos* be the Toby's current position.
- Jump a distance d, d is uniformly distributed over $[1, min(X - pos,\ 10)]$.
- If the new position is the frog's position, catch it and send it as tribute to the queen.
- In other case start the algorithm again.

Note that the length of Toby's jump cannot be infinite, in fact, it must be less than or equal to 10. Besides this, he will never jump over the frog, in other words, he will never reach a position greater than X.

TJ the frog does not want to be catched, due to this, TJ wants to compute the expected number of jumps that Toby needs in order to reach cell number X.

Help to TJ compute this value.

Input

The input starts with an integer $1 < T \leq 100$ indicating the number of test cases.

Each test case contains one integer $10 \leq X \leq 5000$ denoting the frog's cell

Output

For each test case print in one line the expected number of jumps that Toby needs to reach cell number X.

Answers with relative error less than 10^{-6} will be considered correct.

Example

Input	Output
2 10 20	2.9289682540 4.8740191199

Problem I. Sum of all permutations

Source file name:	sumperm.c, sumperm.cpp, sumperm.java
Input:	Standard
Output:	Standard
Author(s):	Sebastián Gómez - UTP Colombia

Toby is very bored because his father went to live to Brazil, so he decided to create a challenge that might take a lot of time to solve. First he creates a function called

SadToby

that receives an array of integers called permutation and a number M as follows:

```
def SadToby(permutation, M):
    sum = 0
    for each x in permutation:
        if (x <= M):
            sum = sum + x
        else:
            break
    return sum
```

For every permutation of the numbers from 1 to N Toby needs to print the sum of SadToby function. Toby needs to compute this result for every possible value of M between 1 and N. As each of this values can be very large output the result modulo the prime $p = 1711276033 = 2^{25} \times 51 + 1$. Can you help this cute dog with his task?

Input

The input consists of several test cases. Each test case begins with a line with one integers N.

- $1 \leq N \leq 10^5$

Output

For each test case, print a single line with N space separated integers containing the required sum for every value of M between 1 and N.

Example

Input	Output

1
2
3
	1
	1 6
	2 9 36

Explication

Third case, first output number $M = 1$. Consider all permutations. If the first number is greater than 1, then the loop will break in the beginning itself with output 0. There are a total of 6 distinct permutations out of which 4 will give 0. The remaining 2 will fetch 1 each from the function. Thus the answer is 2. For $M = 2$ it's easy to check that the output is 9 and for $M = 3$ is 36.

Problem J. Josephus lottery

Source file name:	josephus.c, josephus.cpp, josephus.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Hugo Humberto Morales Peña & Sebastián Gómez - UTP Colombia`

Professor Humbertov Moralov wants to make a raffle between the students of his Data Structure class and Pepito (a student of this group) suggests to use the Josephus problem to determine who is the winner of the raffle. The problem is that you can know beforehand the winning position if you know the value of n (the total of students in the raffle) and the value k (the amount of movements before throwing out a student from the circle).

The prize is kind of interesting, the winner won't have to take the final exam, and for that reason the professor Humbertov proposes the following variant to the Josephus problem: "Take the student class list, in which the students are numbered from 1 to n, then, organize these numbers in a circle and begin to count clockwise from number 1 to the value k. The student with number k in the list is removed from the circle, and now you begin to count, now counterclockwise, from the number of the next student $(k + 1)$. The student with the number in which the count stopped is removed from the circle, and then you repeat the process alternating between clockwise and counterclockwise, counting until you get the winner of the raffle".

Input

The input contains several test cases. Each test case has only one line, in which there are two positive integers N $(1 \leq N \leq 10^6)$ and K $(1 \leq K \leq N)$ that represents respectively, the number of students in the raffle and the value of movements to remove students from the circle. The input ends with a case containing two zeros, which must not be processed.

Output

For each test case you have to print in one line, the number in the student list that represents the winner of the raffle.

Example

Input	Output
`10 1`	`6`
`10 5`	`2`
`10 10`	`5`
`5 5`	`4`
`5 4`	`2`
`0 0`	

Explication

This is the sequence for each step in the case "5 4": $\underline{1}$ 2 3 4 5

1 2 3 4 $\underline{5}$

1 2 3 $\underline{5}$

~~1~~ 2 3 $\underline{5}$

2 3 $\underline{5}$

$\underline{2}$ 3 ~~5~~

$\underline{2}$ 3

$\underline{2}$ ~~3~~

$\underline{2}$ ← The winner

PREMIO FASE I

Problem A. Ambitious journey

Source file name:	ambitious.c, ambitious.cpp, ambitious.java
Input:	Standard
Output:	Standard
Author(s):	Juan Pablo Marín Rosas - CUCEI México

John the explorer is known to travel a lot, on each of his travels he plans always to collect the most souvenirs he can. John is not that greedy, He doesn't want to keep all the souvenirs for himself, his family is huge and He always wants to have enough souvenirs so He can distribute them into his family. On his last travels, he found that the souvenir stores are always ditributed into a square grid of size N, each coordinate of the grid has a store where he can buy up to $S_{i,j}$ souvenirs.

As John is very considered with his family, He always makes the duration of his travels the less time possible so he can spend more time with his family than traveling. To achieve this, He always lands on the coordinate $(1, 1)$ and moves up to the coordinate (N, N), assuming John is in the coordinate (i, j), the next coordinate he will go is either $(i + 1, j)$ or $(i, j + 1)$, also, each time John arrives to a coordinate (including his landing in $(1, 1)$) he will buy all the souvenir available in that store.

John has the maps of the next places He will be traveling. Help John writing a program to calculate for each map the maximum ammount of souvenirs he can buy.

Input

The input consist of several test cases. Each test case begins with a line with a single integer N, followed by N lines with N integer numbers each, where the $i-th$ line and $j-th$ column of the input is the value $S_{i,j}$). The end of the test cases is given by a line where N $=$ 0 , this last line should not be processed as a test case.

- $1 \leq N \leq 1000$

Output

For each test case print in one line the maximum ammount of souvenirs John can get from his travel.

Example

Input	Output

3
1 2 3
1 2 3
1 2 3
4
10 28 12 3
8 25 11 13
15 21 32 10
10 9 8 7
0
	12
	133

Explication

For the first test case John can follow the path $(1, 1), (1, 2), (1, 3), (2, 3), (3, 3)$ to sum up to 12 souvenirs, there is no path where He can get more souvenirs.

Problem B. Building lost

Source file name:	building.c, building.cpp, building.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas - CUCEI México`

On the Amazing City of Mexico (ACM) each street has up to N buildings, each building of the street has a number between 1 and N. If you were walking over the street, you will see that the numeration of the buildings is ordered, this is, the first building has the number 1, the second building has the number 2, and so on.

During the last ACM Programming Contest, John was instructed to give a tour to the foreign contestants who are visiting the town, in this travel he found that all the streets have a missing building, this is, there is a number X between 1 and N that no building in the street has that number, this as you can see, means the street has only $N-1$ buildings not N.

John is preparing a letter to the government where he will ask they to fix this problem, if there are $N-1$ buildings in the street they should numerate the buildings appropriately, He have colected the numbering from T streets. Since some streets have a large amount of buildings, it is difficult for him to find the missing building in all of them, that's why he is requesting your help to write a program that finds the missing building for each street.

Input

The first line of input is a single number T, followed by the description of the T streets. Each street description starts with a line with a single number N followed by a line with $N-1$ numbers showing the numbers the buildings have.

- $1 \leq T \leq 100$
- $1 \leq N \leq 100,000$

Output

For each test case print in one line the number X missing in that street.

Example

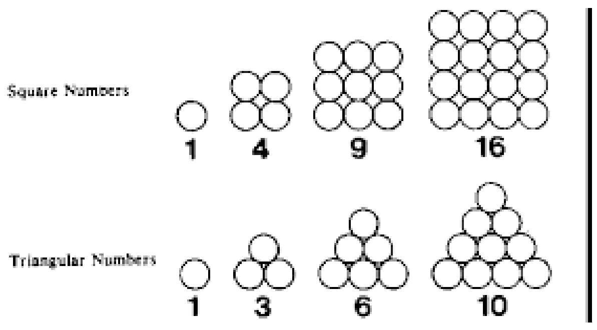
Input	Output

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

Problem C. Counting trapezoids

Source file name:	counting.c, counting.cpp, counting.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas - CUCEI México`

In mathematics there are sets of interesting numbers, some of them have a geometric representation, some examples are the square numbers and the triangular numbers. Square numbers are those that if you had N units you can arrange them in such a way that you can create a square with that units. Triangular numbers are those where the N units can be arranged in such a way that a triangle is created from L consecutive numbers starting from 1.



Some square and triangular numbers

There is another interesting set, we call it the trapezoid numbers, a trapezoid number N, is a number where the units can be arranged in a trapezoid figure from a number of 2 or more consecutive positive numbers, Triangular numbers are also trapezoid numbers that starts counting from 1. An example of trapezoid number is 5 which can be represented as a trapezoid with two numbers {2,3}.

Your task is given a number N , determine how many distinct trapezoids can be drawn using N units?

Input

The input consists of several test cases. Each test case consists of a single line containing a single number N. The end of the test cases is given by the end of file (EOF).

- $1 \leq N \leq 10^9$

Output

For each test case print in one line the number of different ways N can be represented as a trapezoid.

Example

Input	Output
1 3 9	0 1 2

Explication

There are 3 test cases in the file.
For the first test case the output is 0, there is no way to represent 1 as a trapezoid.
For the second test case the output is 1, the only way to represent 3 as a trapezoid is {1,2}
For the third test case the output is 2, there are two ways to represent 9 as a trapezoid :{2,3,4}, {4,5}.

Problem D. Dynamic Writing

Source file name:	dynamic.c, dynamic.cpp, dynamic.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas - CUCEI México`

Only one month to end school and professor teached a new way of writing... It's called dynamic writing, why dynamic? I don't know, but it basically consists on writing words separated by spaces (yes, pretty much as every one writes). Dynamic writing has some rules:

- A letter written with the same length of words separated by the same spaces is considered the same, i.e the letter "There_is_a_way" and "Never_is_a_one" (underscore represents a whitespace) are the same, since the lengths of each word separated by a space is the same.
- The length of a letter is the sum of the length of the words, in the previous example the length is 11 (5+2+1+3).
- There can be words with length 0, this is represented by consecutive whitespaces: "ABC__ABC" has 3 words, two with length 3 ("ABC") and 1 with length 0.
- Leading or trailing whitespaces also separate words "_ABC_" has 3 words, one at the beginning with length 0, then "ABC" with length 3, finally one at the end with length 0.

.

Since you like programming questions, you decided to give you a programming challenge with this new "dynamic writing" thing. Given two values N and K, how many different letters can be written that has length N and exactly K whitespaces?

Input

The input consists of several test cases. Each test case consists of a single line containing two numbers separated by a space N and K. The end of the test cases is given by the end of file (EOF).

- $1 \leq N \leq 10^6$
- $0 \leq K \leq 10^6$

Output

For each test case print in one line the number of different letters that can be written that has length N and exactly K whitespaces modulo $10^9 + 7$;

Example

Input	Output

3 0
3 1
10 3
	1
	4
	286

Problem E. Extended simulation

Source file name:	extended.c, extended.cpp, extended.java
Input:	Standard
Output:	Standard
Author(s):	Juan Pablo Marín Rosas - CUCEI México

Finally the simulation box has arrived, a simulation box has a number of N spots, a number M of links that link these spots, and a ball that will run on the simulation box during a simulation.

Linking on the spots is directional, this means, over a link something can pass only in one direction not both, if you want to do this, then the simulation box requires two different links i.e link $A- > B$ links the spot A with spot B allowing the ball to go from A to B but not from B to A.

The interesting part is running a simulation on the box, for this, the only thing required by human interaction is putting the ball in one of the spots. Once the ball is placed in the spot the ball will begin moving between the links, selecting a link at random that has as source the spot where the ball is currently placed and then moves to the destination of that link, the simulation finishes when the ball reaches the spot where it started.

There is one property in the box that we want to test, we know the ball moves at random but there may be some links that move the ball to one place where even if we run the simulation forever the ball will never reach its initial spot (simulation will be extended, nevec finishes), we call this links "lost links". Since having the simulation running forever won't help on determining wheter or not the ball will reach it's initial state, we ask you to write a program that helps us find given the simulation box configuration (spots and links) answer for each spot asked how many "lost links" can be reached if the simulation is started in that spot.

Input

The input consists of several test cases. The first line contains a single number T, the number of test cases to follow. Each of the T test cases starts with a line containing two integer numbers N and M. The next M lines contains two numbers A and B $(1 \leq A, B \leq N)$ that represents there exists a link from A to B. After the links description there will be a line with a single integer Q ($1 \leq Q \leq N$) the number of spots we are interested on knowing the number of lost links, followed by a line with Q integers each of these is a spot to answer how many "lost links" can be reached if the simulation is started in that spot.

- $1 \leq N \leq 1000$
- $1 \leq M \leq \frac{N(N-1)}{2}$
- $1 \leq Q \leq N$

Output

For each test case print exactly Q lines, the $i-th$ line will have the answer for the $i-th$ spot asked.

Example

Input	Output

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

Explication

There will be only one test case. The test case has a box with 7 spots and 8 links. Then you are asked for 3 spots: 1, 6 and 7. From spot 1 and 6 only one "lost link" can be reached, from spot 7 there are no "list links" reachable.

Problem F. Friendly sum

Source file name:	friendly.c, friendly.cpp, friendly.java
Input:	Standard
Output:	Standard
Author(s):	Juan Pablo Marín Rosas - CUCEI México

John was joking with his friends about how slow all of them sum. Then to improve their sum velocity all the friends decided to play a simple game. At the beggining each of the N friends pick a number, and will get a list of randomly selected friends each friend on the A list will be game peers, note that since each one receive a list, if B is in A list, not necessarily A is in B list.

After all this setup finishes, all the friends will run K rounds, each round consist on summing the numbers of all the game peers (i.e A will sum the number that each of the friends in his list have), all friends will wait for the others to finish summing, and then, when all of them finished, they change their number with the sum they got.

John doesn't like to play this kind of games, so he gave you the number of friends, the number each friend picked at the beginning, the list of each friends game peers and the number of rounds to run. He wants your help to determine what will be the number each friend has when all the K rounds have finished.

Input

The input consists of several test cases. Each test case begins with two numbers N and K. Followed by N lines, each of these lines starts with two numbers P_i which is the number the friend i selected at the beginning and L_i, which is the number of friends on the list of the $i-th$ friend, the rest of the line contains L_i numbers, each is one of the friends in i game peers list.

- $1 \leq N \leq 60$
- $1 \leq K \leq 10^9$

Output

For each test case print N lines. The $i-th$ line contains a single number, the number that the $i-th$ friend has after the K rounds were played. Since this number can be large print the result modulo 10^9+7.

Example

Input	Output

3 2
10 2 2 3
10 2 1 3
10 2 1 2
	40
	40
	40

Explication

There are 3 friends and will play 2 rounds.
Friend 1 picked the number 10 and his peers are the friends 2 and 3.
Friend 2 picked the number 10 and his peers are the friends 1 and 3.
Friend 3 picked the number 10 and his peers are the friends 1 and 2.
On the first round, friend 1 will sum $10 + 10 = 20$.
Friend 2 will sum $10 + 10 = 20$ and friend 3 will sum $10 + 10 = 20$. After all summed, they change their numbers and now Friend 1 has the number 20, friend 2 has the number 20 and friend 3 has the number 20.
On the second and last round, Friend 1 will sum $20 + 20 = 40$, friend 2 will sum $20 + 20 = 40$ and friend

3 will sum $20 + 20 = 40$. After all this they change their numbers and now Friend 1 has the number 40, friend 2 has the number 40 and friend 3 has the number 40.

Problem G. Gatuno's Fiber

Source file name:	base32.c, base32.cpp, base32.java
Input:	Standard
Output:	Standard
Author(s):	Félix Arreola - CUCEI México

The students of Internet Programing are designing several new standards to improve the Internet as we know it. In fact, today have invented a new way of transmitting information thousand times faster than the optical fiber. The new standard name is Gatuno's Fiber.

Oddly, the medium has a small limitation, it can only transmit lowercase letters of the English alphabet (a-z) and the symbols ! @ # $ % & (exclmation mark, at sign, number sign, dollar sign, percent sign, ampersand sign)

One file can have a lot bytes outside the allowed letters (and signs), for this reason, the development team will also use a codification standard called Base32. In this codificaction schema, all bytes converted to binary. Next, are concatenated in a large string of bits. The bits are taken 5 by 5 to form a letter from 0 to 31, which corresponds to a symbol allowed in Gatuno's Fiber as show next:

Bits	Symbol	Bits	Symbol	Bits	Symbol	Bits	Symbol
00000	!	01000	c	10000	k	11000	s
00001	@	01001	d	10001	l	11001	t
00010	#	01010	e	10010	m	11010	u
00011	$	01011	f	10011	n	11011	v
00100	%	01100	g	10100	o	11100	w
00101	&	01101	h	10101	p	11101	x
00110	a	01110	i	10110	q	11110	y
00111	b	01111	j	10111	r	11111	z

For every 5 input bytes, they generate 8 output symbols. When there are not enough bytes to form the 8 symbols, is filled with zeros until a symbol is completed. For example, with 1 input byte, there are 2 output symbols (1 byte has 8 bits, so you need at least 2 symbols, 10 bits, to represent the byte).

Your task is help the students of Internet Programing to write a base32 encoder.

Input

The input consist of integer numbers $0 \leq N \leq 255$, one per line. Each integer represents a byte for encoding to Base32. The end of the bytes is given by the end of file (EOF).

Output

You must print the encoded symbols in base32 for the input bytes, printing a maximum of 80 symbols per line and followed by a newline. The last line must have a newline ending too.

Example

Input	Output
219 232 58 99 46	vjo$osti

Input	Output
132 101 58	klmno

Problem H. Hiding Sequence

Source file name:	hiding.c, hiding.cpp, hiding.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas - CUCEI México`

This problem is easy and fast to read, is it also easy and fast to solve ?

We call a hiding sequence a sequence that the sum of all its elements is equals to 0. As an example the sequence $(1, -1, 2, -2)$ is a hiding sequence but the sequence $(1, -1, 2)$ is not.

Given a list of N numbers, your task is to count how many sequences the list has that are hiding sequences. A sequence on the list can be obtained selecting two positions on the list and taking all the elements between them inclusive. (i.e $(1, -1, 2)$ is a valid sequence from the list $(1, -1, 2, -2)$, but $(1, 2)$ is not).

Input

The input file consists of two lines: The first line contains a single number N. The second line contains N integer numbers, each A_i from the list.

- $1 \leq N \leq 10^6$
- $-10^6 \leq A_i \leq 10^6$

Output

Print a single integer, the number of valid sequences in the list that are a hiding sequence.

Example

Input	Output
5 1 -1 1 -1 1	6

Explication

The valid sequences that are hiding sequences from the list $1, -1, 1, -1, 1$ From 1st to 2nd element : $1, -1$
From 1st to 4th element : $1, -1, 1, -1$
From 2nd to 3rd element : $-1, 1$
From 2nd to 5th element : $-1, 1, -1, 1$
From 3rd to 4th element : $1, -1$
From 4th to 5th element : $-1, 1$

Problem I. Iterating Wheel

Source file name:	iterating.c, iterating.cpp, iterating.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas - CUCEI México`

The iterating wheels is a set of N wheels that will rotate when you push a button. During it's rotation the $i-th$ wheel will pick a token that is positioned just below it on the $i-th$ position and will move it to the P_i position.

It is warrantied that each position is reached only by one wheel and that all wheels reach only one position to move the token. At the beginning token 1 is below wheel 1, token 2 is below wheel 2 and so on.

Given the number of wheels in the iterating wheel, and the position P_i to which each wheel moves the token below it, your task is to determine what is the minimum number of times you have to press the button to leave the tokens in the same position where they started before the first time the button was pressed.

Input

The input file consists of several test cases, the first line for each test case starts with a single integer number N. Followed by one line containing N integers separated by one space, where the $i-th$ number is the value P_i. The end of the test cases is given by the end of file (EOF).

- $1 \leq N \leq 10^6$

Output

For each test case you must print a single number, the number of times you have to press the button to leave the tokens in the same posistion where they started. Since this number can be very large print it modulo $10^9 + 7$

Example

Input	Output

`4`
`2 1 4 3`
	`2`

Explication

Tokens start in the order 1,2,3,4. The first time the button is pressed the tokens will be in the order : 2,1,4,3 The second time the button is pressed the tokens will be in the order : 1,2,3,4 Then, pressing two times the button the tokens will be in the start position.

Problem J. Jacksonville Police Departament

Source file name:	jacksonville.c, jacksonville.cpp, jacksonville.java
Input:	`Standard`
Output:	`Standard`
Author(s):	`Félix Arreola - CUCEI México`

Emergency!

The police in Jacksonville are following a suspect wanted for theft. In fact, he stole the ACM Contest's problem set. Lucky for the police, the thief hide inside a building. The building is surrounded and the police is about to enter.

But, the building has K departments, each identified by a number from 1 to K. After a quick search on the ACM Search Engine, the police found the name of suspect. He has an obsessive compulsive disorder and he only can enter on departments where the number is divisible by one of the thief favorite's numbers.

After another search on ACMBook profile page, they found that the thief has N favorite numbers. The police needs to know how many departments (in the worst case) is going to check.

Help the police to calculate this number in order to recover the lost problem set.

Input

The input consist of several test cases. Each test case consists of a line containing the numbers K and N. After that, there are N lines, each with one of the thief's favorite number F. The end of the test cases is given by 0 0

- $1 \leq K \leq 255$
- $1 \leq N \leq 20$
- $1 \leq F \leq 1000$

Output

For each test case, you should print the maximum number of departments that will be checked.

Example

Input	Output

16 2
3
5
100 1
2
20 3
2
3
4
0 0
	7
	50
	13

TORNEO FASE I

Colombian Collegiate Programming League
CCPL 2015

Round 9 – August 22

Problems

This set contains 12 problems; pages 1 to 18.
All problems in the set are original.

Official site `http://programmingleague.org`

Follow us on Twitter `@CCPL2003`

A - Prove Them All

Source file name: `all.c`, `all.cpp`, *or* `all.java`
Author(s): Camilo Rocha

Alex is a developer at the *Formal Methods Inc.* central office. Everyday Alex is challenged with new practical problems related to automated reasoning. Along with her team, Alex is currently working on new features for a computational theorem prover called "Prove Them All" (or PTA for short). The PTA inference engine is based, mainly, on the *modus ponens* inference rule:

$$\frac{\psi, \quad (\psi \rightarrow \phi)}{\therefore \phi}$$

This rule is commonly used in the following way: for any pair of formulae ϕ and ψ, if there is a proof of the formula ψ and a proof of the logical implication $(\psi \rightarrow \phi)$, then there is a proof of ϕ. In other words, if ψ and $(\psi \rightarrow \phi)$ are theorems, then ϕ is a theorem too.

Today's challenge for Alex and her team is as follows: given a collection of formulae Γ and some relationships among them in the form of logical implication, what is the minimum number of formulae in Γ that need to be proven (outside of PTA) so that the rest of formulae in Γ can be proven automatically using only modus ponens?

Input

The input consists of several test cases. The first line of the input contains a non-negative integer indicating the number of test cases. Each test case begins with a line containing two blank-separated integers m and n ($1 \leq m \leq 10000$ and $0 \leq n \leq 100000$), where m is the number of formulae in Γ of the form ϕ_a and n the number of logical implications which have been proven between some of these formulae. The next n lines contain each two blank-separated integers a and b ($1 \leq a$, $b \leq n$), indicating that $(\phi_a \rightarrow \phi_b)$ is a proven logical implication. Each test case in the input is followed by a blank line.

The input must be read from standard input.

Output

For each test case, output one line with the format "`Case k: c`" where k is the case number starting with 1 and c is the minimum number of formulae in Γ that need to be proven outside of PTA so that the rest of the formulae in Γ can be proven automatically using only modus ponens.

The output must be written to standard output.

Sample Input	Sample Output

```
1
4 4
1 2
1 3
4 2
4 3
```
	`Case 1: 2`

B - Baking Cakes with Grandma

Source file name: `baking.c`, `baking.cpp`, *or* `baking.java`
Author(s): Camilo Rocha

After five years, Eloi is visiting grandma again. She is very proud because he has, by now, mastered the craft of sewing buttons. Eloi is about to finish his degree in math at the Academy of Colombian Mathematicians (ACM); as a matter of fact, he just defended his dissertation about arrangements of colored buttons.

Grandma is currently into baking cakes of all sorts and flavors. "C is for cake; that's good enough for me... fresh from the oven!" Well, for Eloi, C means programming contests, sleepless nights, and humongous cups of Colombian coffee. Anyway, he's here to take a break from that and relax baking cakes with granny.

Grandma usually bakes her cakes in baking pans which were once round, but due to carelessness and time, now have several dents all along their border. Eloi just can't stop thinking about mathematics, so he has realized that there is an interesting geometrical puzzle related to the cakes. Given a circular baking pan with n dents on its border, what is the largest m such that there are m dents amongst the n which form a regular m-gon?

Input

The input consists of several test cases. Each test case consists of two lines. The first line of a test case contains an integer n ($3 \leq n < 10^3$) indicating the number of dents in the border of the baking pan. The second line contains n blank-separated integers $a_1, \ldots, a_n$ (with $1 \leq a_i \leq 10^5$): a_i indicates the length of the arc between the ith dent and the next.

The input must be read from standard input.

Output

For each test case output a line with the largest m such that there are m dents amongst the n that form a regular m-gon. If such an m does not exist, then output "`-1`".

The output must be written to standard output.

Sample Input

```
3
1 1 1
3
1 2 1
4
2 1 1 2
5
2 1 1 2 2
5
1 1 3 1 1
5
1 1 2 1 1
```

Sample Output

```
 3
-1
 3
 4
-1
 3
```

C - Tennis Championship

Source file name: `champion.c`, `champion.cpp`, *or* `champion.java`
Author(s): Rafael García

A certain tennis championship with P players has a particular set of rules:

1. Before every round, players are paired randomly.
2. Each pair so defined establishes a match that will be played.
3. The winner of a match advances to the next round in the tournament and the loser is eliminated from competition.
4. If the number of players before a round is odd, then one player (chosen at random) is automatically promoted to the next round.

This process should be repeated over and over again until there is exactly one player left. Such a player will be the champion.

The Tennis Championship Organization wants to calculate the total number of matches needed to determine the champion.

Input

The input consists of several test cases, each one consisting of a single line containing a positive integer P, the number of players.

The input must be read from standard input.

Output

For each test case, output a line with one integer indicating the number of matches needed to determine the champion.

The output must be written to standard output.

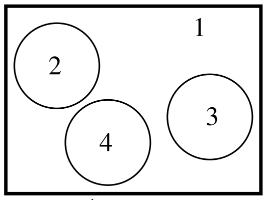
Sample Input	Sample Output
3 2	2 1

D - Euler Diagrams

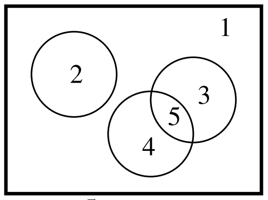
Source file name: `diagrams.c`, `diagrams.cpp`, *or* `diagrams.java`
Author(s): Federico Arboleda, Rafael García, and Alejandro Sotelo

An *Euler diagram* (named after Leonhard Euler) consists of simple closed curves in the plane, usually circles, that depict sets. The spatial relationships between the regions bounded by each curve (overlap, containment or neither) corresponds to set-theoretic relationships (intersection, subset and disjointness, respectively); depending on the relative location and size of the curves, the plane (or, as is usually the case, a paper sheet) is divided in a certain number of *zones*, each one of which represents an intersection of the original sets or their complements. A more restrictive form of Euler diagrams are *Venn diagrams*, which must include all logically possible zones of overlap between its curves.

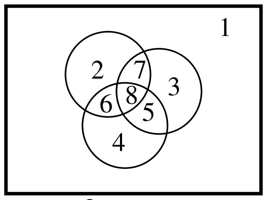
Formally, given circular regions $S_1, S_2, \ldots, S_n$ in the plane, we shall define a *zone* as a nonempty set of the form $f_1(S_1) \cap f_2(S_2) \cap \cdots \cap f_n(S_n)$, where, for each i, either $f_i(S_i) = S_i$ or $f_i(S_i) = S_i{}^c$ (the complement of S_i with respect to the drawing surface).



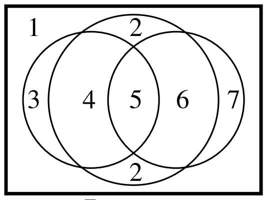
4 zones



5 zones



8 zones



7 zones

Given a rectangular drawing surface and a collection of circles, find the number of zones in which the surface is split. Note that, in the last example, zone 2 is labeled twice even though both labels are in the same set.

Input

The input consists of several test cases. Each case begins with three blank-separated positive integers, W, H and n, which represent, respectively, the width of the drawing surface, the height of the drawing surface, and the number of circles in the diagram ($1 \leq W \leq 1000$, $1 \leq H \leq 1000$ and $0 \leq n \leq 100$). Each one of the next n lines consists of three blank-separated positive integers, x, y and r, specifying the center (x, y) and radius r of a circle ($0 \leq x \leq W$, $0 \leq y \leq H$, and $1 \leq r \leq W + H$).

You may assume every circle is fully contained within the drawing surface, that no two circles intersect at a single point, that every two circles are different, and that the sides of the surface are not tangent to any circle.

The end of the input is given by $W = H = n = 0$, which should not be processed as a test case.

The input must be read from standard input.

Output

For every test case print a line with the number of zones in which the drawing surface was split by the circles.

The output must be written to standard output.

Sample Input

```
60 44 3
12 14 10
24 32 10
48 26 10
60 44 3
16 16 10
34 30 10
44 22 10
60 44 3
24 16 10
28 28 10
36 20 10
60 44 3
30 22 20
20 22 16
40 22 16
50 50 4
25 25 5
25 25 10
25 25 15
25 25 20
50 50 3
25 25 5
25 25 10
25 25 15
50 50 2
25 25 5
25 25 10
50 50 1
25 25 5
50 50 0
50 50 5
15 25 6
20 25 6
25 25 6
30 25 6
35 25 6
50 50 3
25 35 10
15 25 9
35 25 9
0 0 0
```

Sample Output

```
4
5
8
7
5
4
3
2
1
13
6
```

E - Going Shopping with Grandma (I)

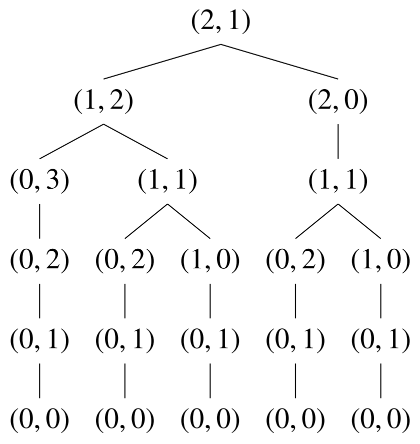
Source file name: `eloi.c`, `eloi.cpp`, *or* `eloi.java`
Author(s): Camilo Rocha

Sometimes, going shopping with grandma can be a very exciting and fun adventure! Eloi is going shopping with grandma this evening because of the holidays; just perfect for his saying: "Sewing, baking, and shopping with grandma, it all goes together… a grandmother, at holiday time, is worth gold." They also are stopping at the pharmacy: granny is losing her memory and her bottle of memory pills is running low ... how sad!

The memory pills come in two sizes: *large* and *small*. The dose in each large pill is equivalent to that in two small ones. Eloi observes granny picks a pill at random from the bottle every day: if it's a small one, she takes it; otherwise, she splits it and takes a half, replacing the other which is from then on considered a small pill.

Given a certain bottle with l large pills and s small pills, we say that the pair (l, s) is the *bottle configuration*. Eloi is interested in the *pill tree* associated with bottle configuration (l, s), in which left or right branching represents a large or small pill being picked, respectively. Formally it's the labeled binary tree with root (l, s) in which a node (u, v) has a *left child* $(u - 1, v + 1)$ if $u > 0$ and a *right child* $(u, v - 1)$ if $v > 0$.

For example, the pill tree associated with bottle configuration $(2, 1)$ (2 large, 1 small) is depicted below:



Eloi then asks himself: how many nodes does the pill tree associated with bottle configuration (l, s) have?

Input

The input consists of several test cases. Each test case consists of a line with two blank-separated integers l and s $(0 \leq l \leq 1000$ and $0 \leq s \leq 1000)$.

The end of the input is given by $l = s = 0$, which should not be processed as a test case.

The input must be read from standard input.

Output

For each l and s, output a line with the number of nodes in the pill tree associated to (l, s). Since this number can be very large, print it modulo 9 999 959 999.

The output must be written to standard output.

Sample Input	**Sample Output**
2 1	21
6 5	31654
100 2	5306431377
19 78	1942584859
1000 1000	4124225148
0 0	

F - Going Shopping with Grandma (II)

Source file name: `pharmacy.c`, `pharmacy.cpp`, *or* `pharmacy.java`
Author(s): Camilo Rocha

After leaving the pharmacy with grandma, Eloi has realized there are still some interesting mathematical puzzles regarding granny's pill taking routine.

Granny's memory pills come in two sizes: *large* and *small*. The dose in each large pill is equivalent to that in two small ones. Eloi observes granny picks a pill at random from the bottle every day: if it's a small one, she takes it; otherwise she splits it and takes a half, replacing the other which is from then on considered a small pill.

Eloi would like to solve the following puzzles regarding a given bottle with l large pills and s small pills:

1. What is the expected number of small pills remaining when the last large pill is picked?
2. What is the expected day in which the last large pill is picked?

Your task is to help Eloi solve those puzzles.

Input

The input consists of several test cases. Each test case consists of a line with two blank separated numbers l and s ($0 \leq l \leq 100$ and $0 \leq s \leq 100$).

The end of the input is given by $l = s = 0$, which should not be processed as a test case.

The input must be read from standard input.

Output

For each test case, output a line with two blank-separated numbers a_1 and a_2: a_1 is the answer to question 1 and a_2 to question 2 above. Each a_i must approximate the correct answer to within 10^{-6}.

The output must be written to standard output.

Sample Input	Sample Output
`2 1`	`1.833333333333 3.166666666667`
`6 5`	`3.164285714286 13.835714285714`
`100 2`	`5.207179497838 196.792820502162`
`19 78`	`7.447739657144 108.552260342856`
`0 0`	

G - Trading Card Game

Source file name: `game.c`, `game.cpp`, *or* `game.java`
Author(s): Federico Arboleda and Alejandro Sotelo

Little Ricky is obsessed over the new trading card game, *Sorcery: The Meeting*. He just cannot stop talking about it! He is so obsessed, in fact, that he spends most of his monthly allowance in buying *Sorcery: The Meeting* trading cards, in the hopes of getting all of them.

Of course, buying every single opened card would be too expensive for poor little Ricky, who has to buy everything on his mother's allowance. Instead, he decides to save up during some months and then buy as many unopened cards as he can, hoping he can get them all. Fortunately for Ricky, unopened cards are sold individually!

Being as obsessed as he is, he knows exactly how many cards are in circulation and that, unlike in some other trading card games, there is exactly the same chance to find each card in any single buy. Sadly, Ricky is not very good at math and so he cannot even begin to comprehend what he's going to find in such a big buy. He has, though, put aside some of his allowance this month to ask for your help in calculating the odds of making a good buy (in exchange for the price of a couple of *Sorcery: The Meeting* cards, of course).

If Ricky tells you there are N *Sorcery: The Meeting* trading cards in circulation, all of them equally likely, and he has saved enough to buy m of them at the same time, what is the probability that he will get exactly k different cards?

Being as obsessed as he is, he knows that floating-point numbers would necessarily incur a loss of precision, which he will not tolerate. Therefore, he wants this information as a fraction in lowest terms.

Input

The input consists of several test cases. Each case is a line with three blank-separated integers, N, m and k, which represent, respectively, the total number of cards in circulation, the number of cards Ricky is going to buy, and the number of different cards he expects to get ($1 \leq N \leq 100$, $0 \leq m \leq 100$ and $0 \leq k \leq 100$).

The end of the input is given by $N = m = k = 0$, which should not be processed as a test case.

The input must be read from standard input.

Output

For every test case print a line of the form "`p/q`", where $\frac{p}{q}$ is a fraction in lowest terms representing the probability that Ricky will get exactly k different cards under the described conditions. A probability of 0 should be represented as "`0/1`" and a probability of 1 should be represented as "`1/1`".

The output must be written to standard output.

Sample Input	**Sample Output**

```
10 1 1
10 2 2
10 12 11
10 4 2
10 4 1
10 4 0
0 0 0
```
	```
	1/1
	9/10
	0/1
	63/1000
	1/1000
	0/1
	```

H - Harvest Moon

Source file name: `harvest.c`, `harvest.cpp`, *or* `harvest.java`
Author(s): Federico Arboleda, Rafael García, and Alejandro Sotelo

The farmer Yasuhiro has recently bought a rectangular plot. Prior to sowing time, the day before a full moon, he traced a rectangular grid on his plot and separated his seeds in two categories: medicinal plants and fruit trees.

The land quality is, of course, not uniform. Some cells of the grid are better than others depending on certain factors such as soil permeability, coarseness, irrigation type, and ground slope. To quantify all those variables, Yasuhiro has defined a *productivity factor* for each cell: a number between 0 and 100 indicating how much he would benefit from sowing in that particular cell. Specifically, a factor of 0 means he would not benefit at all from that cell, while a factor of 100 means he would benefit the most.

Yasuhiro has also made a table with information about the plant species he is going to sow. This table describes the category of each species (medicinal plant or fruit tree), its cost per cell, and the minimum and maximum possible number of cells occupied by that species. The total benefit from sowing one particular species in any particular cell is equal to the productivity factor of that cell times the cost per cell of that species.

Given the number of rows and columns in the grid, the productivity factor of each cell, and the table with the plant information, you must calculate the maximum possible benefit which can be obtained by sowing according to the following rules:

- In each cell, at most one plant species can be sown.
- For every species, the number of cells where it is sown must be between the minimum and the maximum specified in the table.
- No two cells in the same row can contain the same species of medicinal plant. Likewise, no two cells in the same column can contain the same species of fruit tree.
- The total benefit is the sum of the individual benefits from each sown cell.

Input

The input consists of several test cases. The specification of each test case follows:

- First, there is a line with three integers R, C, and E, which specify, respectively, the number of rows in the grid, the number of columns in the grid, and the number of species in the table ($1 \leq R \leq 4$, $1 \leq C \leq 4$ and $1 \leq E \leq 10$).
- Then follow R lines, each one of them with C blank-separated integers between 0 and 100. The jth number of line i is the productivity factor of the cell in row i and column j.
- Finally, there are E lines, one for each species in the table, each comprising the following blank-separated data:
  - The character '`M`' if it's a medicinal plant or '`F`' if it's a fruit tree.
  - An integer d which indicates the current species' cost per cell ($1 \leq d \leq 10^4$).
  - Two integers n and m which are, respectively, the minimum and maximum number of cells for the current species ($0 \leq n \leq m \leq 4$).

The end of the input is given by $R = C = E = 0$, which should not be processed as a test case.

The input must be read from standard input.

Output

For each test case, print a line with the maximum possible benefit which Yasuhiro can obtain from sowing the plot according to the rules. If it is not possible to sow the plot as specified, then print "0".

The output must be written to standard output.

Sample Input	Sample Output
2 2 1 10 10 10 10 M 3000 0 4 3 3 2 10 50 90 50 90 10 90 10 50 M 1500 3 3 F 500 3 3 2 2 1 100 100 100 100 F 2000 3 4 2 3 1 100 100 100 100 100 100 M 2000 0 3 0 0 0	60000 480000 0 400000

I - Accelleratii Incredibus

Source file name: `incredibus.c`, `incredibus.cpp`, *or* `incredibus.java`
Author(s): Federico Arboleda, Rafael García, and Alejandro Sotelo

If you're on a highway and Road Runner goes "beep beep"
just step aside or you might end up in a heap.
Road Runner, Road Runner runs on the road all day.
Even the Coyote can't make him change his ways.
— *The Road Runner Show* theme song —

The Road Runner *(Accelleratii incredibus)* is a very fast-running ground bird which can be found in the roads of the hot and lonely Southwestern United States. Wile E. Coyote *(Carnivorous vulgaris)*, a clever canine, has repeatedly and unsuccessfully tried to catch it using every kind of trap imaginable.

The Interstate is the Road Runner's favourite road, since it's a straight, L miles long path along which it can run at constant speed without stopping or turning. One particular summer day, the Road Runner is sunbathing on the Interstate and wants to run home (also on the Interstate) in exactly m minutes, never exceeding v miles per minute. The Coyote meanwhile has installed bombs at certain positions along the road and programmed them to go off at certain times in the hopes of catching the Road Runner in one of the explosions.

Every minute, the Road Runner can move an integer amount of miles which must be less than or equal to v, avoiding the spots in the road which have a bomb programmed to go off during that minute. Given the positions of the bombs which will go off each minute, you must calculate the minimum amout of miles the Road Runner must move to go from its sunbathing spot to its home in exactly m minutes, avoiding all explosions.

Input

The input consists of several test cases. Each case begins with a line with a positive integer L which is the length of the Interstate in miles ($1 \leq L \leq 1000$). Then follows a line with two integers x_i and x_f which are, respectively, the initial position of the Road Runner and the location of its home ($0 \leq x_i \leq L$, $0 \leq x_f \leq L$). The next line contains two integers m and v which represent, respectively, the amount of minutes in which the Road Runner wants to reach its home and the maximum allowed velocity in miles per minute ($1 \leq m \leq 100$, $1 \leq v \leq L$). Each one of the next m lines contains a string of $L + 1$ characters; character i of line j is '`X`' if there is a bomb at position i which will go off during the jth minute, or '`.`' otherwise.

You may assume that no bomb will go off at position x_i during the first minute. The end of the input is given by $L = 0$, which should not be processed as a test case.

The input must be read from standard input.

Output

For each test case output a line with the minimum amount of miles the Road Runner must move to go home in exactly m minutes and avoiding all explosions. If it is not possible for the Road Runner to go home avoiding all explosions, then output -1.

The output must be written to standard output.

Sample Input	**Sample Output**

```
6
0 6
5 6
...X...
XX..XX.
...X...
.XX....
.......
6
0 6
5 5
...X...
XX..XX.
...X...
.XX....
.......
6
1 6
3 3
.......
...X...
.......
6
1 6
3 2
.......
...X...
.......
0
```
	```
	 10
	-1
	 5
	-1
	```

J - Ant-Man's Sugar Journey

Source file name: `journey.c`, `journey.cpp`, *or* `journey.java`
Author(s): Federico Arboleda and Diego Satoba

Ant-Man is the latest superhero on the stage. Like most superheroes, he has got unique powers – besides shrinking to the size of an insect, he can control ants through his suit.

As usual, Ant-Man should use his powers only for the greater good (e.g., world peace, saving mankind, or supervising programming contests), but more often than not, he uses them to impress his girlfriend, Wasp.

This time he has invited Wasp over for some Colombian coffee. Not being used to the strong taste, she wants to sweeten it, but this being a programming contest, there's one little problem: the ants have taken all the sugar to their nest while Ant-Man wasn't looking.

The ants' nest is unlike any other: it has got only one entrance and one (different) exit, and comprises a network of tunnels connecting them, with several intersections and branching. Since there are many ants living in the tunnels, every tunnel may be run only in a predetermined direction, and there is no path of tunnels from any intersection to itself, no dead ends, and no inaccessible intersections. The ants are keeping a sugar cube at every intersection.

Wanting to impress Wasp some more, he will use his ants to bring his sugar back instead of going in himself, and he will do it with the minimum possible number of ants. Each ant is strong enough to carry an unlimited amount of sugar cubes at the same time, but Ant-Man doesn't want them to feel like tools, so he will not order any ant to re-enter the nest after its sugar run is done.

Ant-Man has asked his old cellmate for help in performing this task, and as usual, he "knew someone who knows someone" who has relayed this problem to you. Now you must calculate the minimum possible number of ants needed to bring back all the sugar.

Input

The input consists of several test cases. Each case begins with a line containing two blank-separated integers N and M ($2 \leq N \leq 100$ and $1 \leq M \leq 5000$), which represent the number of intersections and the number of tunnels in the nest, respectively; the entrance and exit points are counted as intersections. Next come M lines with two blank-separated integers u and v ($0 \leq u \leq N-1$, $0 \leq v \leq N-1$, and $u \neq v$), meaning that there is a tunnel from intersection u to intersection v (running in that direction).

The end of the input is given by $N = M = 0$, which should not be processed as a test case.

The input must be read from standard input.

Output

For each test case print a line containing the minimum number of ants needed to recover all the sugar.

The output must be written to standard output.

Sample Input	**Sample Output**

```
1
9 12
0 1
0 2
1 3
1 4
2 4
2 5
3 6
4 6
4 7
5 7
6 8
7 8
0 0
```
	`3`

K - Prime Kebab Menu

Source file name: `kebab.c`, `kebab.cpp`, *or* `kebab.java`
Author(s): Rafael García

Prime is an unusual but efficient Kebab restaurant. *Prime* has many small tables, which allow it to serve groups of various sizes without problems. We went to *Prime* last weekend and saw one small table with 3 people, another one with 5 people, and another with 14 people – the servers simply rearrange the tables as they see fit to cater to those groups.

One unusual feature of *Prime* is that each client may choose only one dish from their large menu. Even more strangely, we noticed that the waiters of *Prime* relay a whole order to the kitchen with just one number! I saw the group of 3 people ask for the first, third, and fifth dishes in the menu, and their waiter relayed this to the kitchen as 110. When the group of five asked for dishes number 1, 2, 3, 4, and 5, their waiter said only 2310. Finally, when the group of 14 requested the first dish 10 times and the second one 4 times, I heard their waiter say 82 944. Amazingly, when the respective waiters came back with the meals, all orders were correctly fulfilled!

This strange numerical code is still incomprehensible to me. I have found, through some research, that the waiters never say 1. Of course this doesn't help me much and I still have lots of questions, such as: if a waiter relays just one number to the kitchen, how do the chefs calculate the number of clients in the table? I am confident that, if you can answer this for me, then I can fully crack the restaurant code.

Your task is to write a program that outputs the number of clients in the table given the number relayed by a waiter.

Input

The input consists of several test cases. Each test case is a line containing an integer $1 < n < 10^{14}$, which is a number relayed by a waiter.

The end of the input is given by $n = 1$, which should not be processed as a test case.

The input must be read from standard input.

Output

For each test case, print a line with the number of clients in the group.

The output must be written to standard output.

Sample Input	Sample Output
110 82944 1	3 14

L - The Weakest Link

Source file name: `link.c`, `link.cpp`, *or* `link.java`
Author(s): Camilo Rocha

It is clearly a literal fact that a chain is only as strong as its weakest link. The conversion of that notion into a figurative phrase was established in the language by the 18th century. Thomas Reid's *Essays on the Intellectual Powers of Man* (1786), included this line:

> In every chain of reasoning, the evidence of the last conclusion can be no greater than that of the weakest link of the chain, whatever may be the strength of the rest.

In this problem a *chain of length n* is a string $C = c_1c_2\ldots c_n$ of n lowercase characters where c_n is considered to be followed by c_1 in a cyclical fashion. Character i is said to be *weaker* than character j in a chain C if the string $c_ic_{i+1}\ldots c_nc_1\ldots c_{i-1}$ comes before the string $c_jc_{j+1}\ldots c_nc_1\ldots c_{j-1}$ in lexicographical order.

Given a chain C, your task is to find the weakest character in C.

Input

The first line of the input contains a non-negative integer N indicating the number of test cases. Each test case comprises a single line with a nonempty string C of at most 50 000 lowercase characters of the English alphabet '`a`'-'`z`'. You may assume that $\texttt{a} < \texttt{b} < \cdots < \texttt{z}$ as usual.

The input must be read from standard input.

Output

For each test case, output a line containing an integer w such that c_w is the weakest character in the chain C. If there is more than one such value, then output the smallest one.

The output must be written to standard output.

Sample Input	Sample Output

```
4
ccpl
abracadabra
hocuspocus
aa
```
	```
	1
	11
	8
	1
	```

GRAN PREMIO FASE II

acm International Collegiate Programming Contest 2015 IBM event sponsor

Gran Premio de México & Centroamérica

Fase II 2015

September 12th, 2015

Important instructions

1. In each problem, each input file contains only one test case. Your solution will be executed with more than one input file.
2. Each execution of your solution will be limited by time, with a value specified by problem, as indicated in the table below.
3. To consider your solution as correct, it must end with out errors, producing the expected output, in the time limit, for each of the input files.
4. If your solution gives an error or exceeds the time limit for a any input file, you will receive and indication of error (time limit exceeded, wrong answer, presentation error) for that file and the execution will end. The file that caused the error is not identified.
5. Note that there might be other errors, of other kind, in the input file that caused the error or in other different input file, but only the first error found is reported.
6. For solutions in C and C++, it is important that the execution ends with the instruction "`return 0;`", since that indicates to the OS that the execution ended without errors. For solutions in Java the workspace handles the end of execution.
7. The time limits for execution are:

Problem	Name	C/C++	Java
A	Even Obsession	1s	1s
B	Stock Market	1s	3s
C	Tri-du	1s	1s
D	Puzzle	1s	3s
E	Spiral	1s	1s
F	Factorial	1s	1s
G	Curious Guardians	1s	2s
H	Rectangle Park	1s	2s
I	Ominobox	10s	13s
J	Strategy Game	1s	3s
K	Palindrome	1s	4s
L	Lottery	1s	3s

8. Execution information:

Compiling Commands

```
C: gcc -static -O2 -lm
C++:  g++ -static -O2 -lm
C++11:  g++ -std=c++11 -static -O2 -lm
Java:  javac
```

Command for Java execution

```
java -Xms512m -Xmx512m -Xss51m
```

Available memory for programs in C/C++

512 MB

acm International Collegiate Programming Contest 2015 IBM event sponsor

Gran Premio de México & Centroamérica

Fase II 2015

September 12th, 2015

Problem Book

General Information

This book contains 12 problemas; pages are numbered. from 1 to 15, excluding this cover page. Verify your copy is complete.

A) About the program names
1) Your code must me named *codigo_de_problema*`.c`, *codigo_de_problema*`.cpp` or *codigo_de_problema*`.java`, where *codigo_de_problema* is the upper case letter that identifies the problem. Remember that in Java, the name of the principal class is the same than the name of the file.

B) About the Input
1) The input of you program must be read from *standard input.*
2) The input is formed by only one test case, described in the number of lines that depends in the problem.
3) When one line contains more than one value, this are separated by one unique blank space; the input does not contain more than one blank space.
4) Each line, including the last line, contains exactly one end-of-line character.
5) The end of the input matches with the end of file.

C) About the Otput
1) The output of your program must be printed to *standard output.*
2) When one line contains more than one values, these must be separated by one unique blank space; the output must not contain any other blank space.
3) Each line, including the last line, must contain exactly one end-of-line character.

Problem A
Even Obsession

Patricia is an excellent software developer, but, as every brilliant person, she has some strange quirks. One of those is that everything she does has to be in even quantities. Most often that quirk does not affect her, even though it may seem strange to others. Some examples: every day she has to eat an even number of meals; during breakfast, she drinks two cups of coffee, eats two toasts and two slices of cheese; when she goes to the cinema she buys two tickets (fortunately she always has a friend that goes with her); she takes two baths per day (or four, our six...).

Some other times, however, that quirk makes the life of Patricia more difficult. For example, no one wants to travel by car with her because if she has to pay toll, the number of tolls she pays has to be an even number.

Patricia lives in a country where all roads are two-way and have exactly one toll each. She needs to visit a client in a different city, and wants to calculate the minimum total value of tolls she has to pay to go from her city to the client's city, obeying her strange quirk that she has to pay an even number of tolls.

Input

The input consists of several test cases. The first line of a test case contains two integers C and V, the total number of cities and the number of roads ($2 \leq C \leq 10^4$ and $0 \leq V \leq 50000$). The cities are identified by integer numbers from 1 to C. Each road links two different cities, and there is at most one road between each pair of cities. Each of the next V lines contains three integers C_1, C_2 and G, indicating that the toll value of the road linking cities C_1 and C_2 is G ($1 \leq C_1, C_2 \leq C$ e $1 \leq G \leq 10^4$). Patricia is currently in city 1 and the client's city is C.

Output

For each test case in the input your program must output exactly one line, containing exactly one integer, the minimum toll value for Patricia to go from city 1 to city C, paying an even number of tolls, or, if that is not possible, the value -1.

Examples

Input	**Output**

```
4 4
1 2 2
2 3 1
2 4 10
3 4 6
5 6
1 2 3
2 3 5
3 5 2
5 1 8
2 4 1
4 5 4
```
	```
	12
	-1
	```

Problem B
Stock Market

A beginner investor wants to learn how to invest in the stock market. As he does not have any experience, he selected one company and followed daily the value of the stock during N days. At the end, he wondered how much money he would have won if he had invested during the time he followed the stock value. To be honest, the investor is multi-billionaire and has a lot of money, enough to buy any amount of stock actions of the company. However, as he is very careful with his investments, he decided that he would never have more than one stock of the company.

To cover his costs, the stockbroker charges a fixed rate of C dollars for every stock purchase.

You have to calculate the maximum profit that the investor could have won investing during the N days, having also the option of not to invest any money.

Input

The input consists of several test cases. The first line of a test case contains two integers, N and C ($1 \leq N \leq 2 \times 10^5$, $0 \leq C \leq 30$). The second line contains the N prices $P_1, P_2, \ldots, P_N$ of the days $1, 2, \ldots, N$, respectively. Every price P_i satisfies $1 \leq P_i \leq 1000$.

Output

For each test case in the input your program must produce exactly one line, containing exactly one integer, the maximum profit of the investor, in dollars.

Examples

Input	Output

```
6 10
100 120 130 80 50 40
5 10
70 80 50 40 50
13 30
10 80 20 40 30 50 40 60 50 70 60 10 200
```
	```
	20
	0
	220
	```

Problem C
Tri-du

Tri-du is a card game inspired in the popular game of Truco. The game uses a normal deck of 52 cards, with 13 cards of each suit, but suits are ignored. What is used is the value of the cards, considered as integers between 1 to 13.

In the game, each player gets three cards. The rules are simple:

- A Three of a Kind (three cards of the same value) wins over a Pair (two cards of the same value).
- A Three of a Kind formed by cards of a larger value wins over a Three of a Kind formed by cards of a smaller value.
- A Pair formed by cards of a larger value wins over a Pair formed by cards of a smaller value.

Note that the game may not have a winner in many situations; in those cases, the cards are returned to the deck, which is re-shuffled and a new game starts.

A player received already two of the three cards, and knows their values. Your task is to write a program to determine the value of the third card that maximizes the probability of that player winning the game.

Input

The input contains several test cases. In each test case, the input consists of a single line, which contains two integers A ($1 \leq A \leq 13$) and B ($1 \leq B \leq 13$) that indicates the value of the two first received cards.

Output

For each test case in the input, your program must produce a single line, containing exactly one integer, representing the value of the card that maximizes the probability of the player winning the game.

Examples

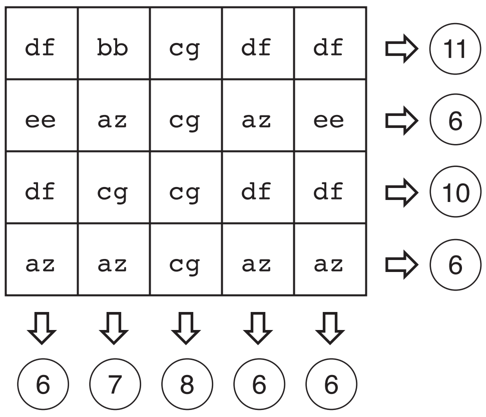
Input	Output

`10 7`
`2 2`
	`10`
	`2`

Problem D

Puzzle

Recent discussions in the Internet caused a renewed interest in logical puzzles. In this problem your task is to write a program that solves puzzles as the one shown in the figure below. In this kind of puzzle, the letters inside the grid represent variables, and the numbers represent the sum of the values of the variables in each line or column.



The objective of this type of puzzle is to determine the value of each variable so that the sum of the lines and columns shown is satisfied. But as this type of puzzle is intended for kids, it has a property that makes it easier to solve: it is always possible to find a line or column which contains only one variable whose value is not yet known. So, one way to solve the problem is to proceed step by step, finding the value of one variable at each step.

Given a puzzle, you need to determine the values of the variables that will solve it.

Input

The input consists of several test cases. The first line of each test case contains two integers L ($1 \leq L \leq 100$) and C ($1 \leq C \leq 100$) indicating the number of lines and the number of columns in the puzzle. Each of the next L lines contains C names of variables, followed by an integer S, the sum of the variables in that line ($-10^8 \leq S \leq 10^8$). The last line contains C integers X_i ($-10^8 \leq X_i \leq 10^8$), indicating respectively the sum of the variables in column i. The names of the variables are formed by exaclty two lower case letters, from 'a' to 'z'. All puzzles have a unique solution, in which all variables are integer numbers between -10^6 and 10^6.

Output

For each test case, your program must produce one line for each variable of the puzzle, containing the name of the variable and its integer value. The variables should be produced in ascending alphabetical order; in other words, respecting order

$$\texttt{aa}, \texttt{ab}, \ldots, \texttt{az}, \texttt{ba}, \texttt{bb}, \ldots, \texttt{za}, \texttt{zb}, \ldots, \texttt{zz}.$$

Examples

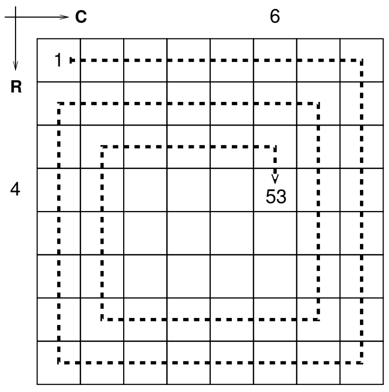
Input

```
4 5
df bb cg df df 11
ee az cg az ee 6
df cg cg df df 10
az az cg az az 6
6 7 8 6 6
3 4
aa bb cc dd 10
aa bb cc dd 10
aa bb cc dd 10
3 6 9 12
3 3
aa zz aa 27
vv zz aa -5
kk kk aa 40
15 -7 54
```

Output

```
az 1
bb 3
cg 2
df 2
ee 1
aa 1
bb 2
cc 3
dd 4
aa 18
kk 11
vv -14
zz -9
```

Problem E
Spiral



Given a $N \times N$ grid, we would like to place beans, one in each square, following a spiral as shown in the picture. Starting from the upper-left square, with coordinates $(1, 1)$, and then going to the right until possible, then down until possible, then left until possible and then up until possible. We repeat this pattern, right-down-left-up, until B beans are placed into the grid. The problem is: given N and B, at which coordinates will the last bean be placed? In the picture, for $N = 8$ and $B = 53$, the last bean is placed at coordinates $(4, 6)$.

Input

The input contains several test cases. A test case consists of a single line containing two integers, N and B, where $2 \leq N \leq 2^{30}$ and $1 \leq B \leq N^2$.

Output

For each test case in the input your program must output one line containing two integers, R and C, where (R, C) are the coordinates of the last bean.

Examples

Input	Output

`8 53`
`1073741824 1152921504603393520`
	`4 6`
	`536871276 536869983`

Problem F
Factorial

The *factorial* of a positive integer number N, denoted as $N!$, is defined as the product of all positive integer numbers smaller or equal to N. For example $4! = 4 \times 3 \times 2 \times 1 = 24$.

Given a positive integer number N, you have to write a program to determine the smallest number k so that $N = a_1! + a_2! + \ldots + a_k!$, where every a_i, for $1 \leq i \leq k$, is a positive integer number.

Input

The input consists of several test cases. A test case is composed of a single line, containing one integer number N ($1 \leq N \leq 10^5$).

Output

For each test case in the output your program must output the smallest quantity of factorial numbers whose sum is equal to N.

Examples

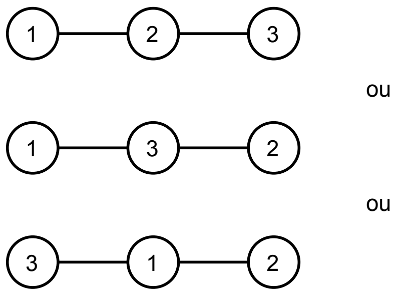
Input	Output

`10`
`25`
	`3`
	`2`

Problem G
Curious Guardians

Oa is one of the most antique worlds of the DC universe, and it is the home of the guardians of the universe. They manage the Green Lantern troop, one of the major forces of the universe! Everyone knows that the Green Lanterns can fly because they have the power of the ring. However, not all inhabitants of Oa are part of the troop. For those inhabitants that do not fly it is difficult to move between cities, because there are no highways.

The guardians want to connect the cities of Oa building highways. There are N cities in Oa, and they plan to build $N-1$ two-way highways, so it is possible to go from any city to any other city, directly or indirectly. The guardians do not want to give more privileges to any city, so they have established that no city can have more than K highways. For example, if we have three cities and K equals 2, we have three options:



The guardians, who are very curious, asked the Green Lanterns if they are capable of telling in how many ways the $N-1$ highways can be built following these restrictions. Your task, as a member of the Green Lantern troop is, given N and K, answer the guardian's question.

Input

The input consists of several test cases. A test case consists of one line containing two integers N $(1 \leq N \leq 10^2)$ and K $(1 \leq K \leq N)$.

Output

For each test case in the input your program must produce one single line with one single integer number, the answer to the problem. As the answer can be very big, output it modulo $10^9 + 7$.

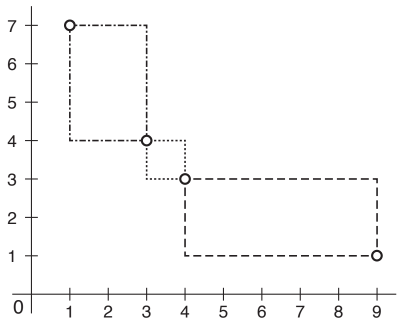
Examples

Input	**Output**
`3 2`	`3`
`4 1`	`0`
`4 3`	`16`

Problem H
Rectangle Park

Rectangleland is a very old city that has kept several historical treasures. The city was planned many decades ago, with all the roads going north-south or east-west. At the moment, there is a project to revitalize the city, which plans to build a new rectangular park. The public administration will choose the location of the park and for the moment they are interested in the possible locations for the park, considering that it needs to be aligned to the roads. That is, when is visualized in a map, the borders of the park should be horizontal or vertical. With the objective of preserving the historical treasures when building the park, some restrictions must be obeyed.

In the city there are some lampposts from the XIX century. Given their historical value, no lamppost must be removed during the construction of the new park. Due to natural decay and bad maintenance, few historical lampposts remain, and no road has more than one. No historical lamppost should be inside the park. On the other hand, the landscape project plans to have two historic lampposts in two corners of the park. The figure below shows an example of four historic lampposts and the three possible locations for the park.



The city hall hired a georeference company to map the lampposts positions. With this data at hand, the next step is to determine how many possible locations exist for the park, so the size of the teams necessary to evaluate each location can be estimated.

Input

The input contains several test cases. The first line a test case contains an integer number N, $1 \leq N \leq 3000$, which represents the number of historical lampposts. The next N lines will describe, each, the position of a lamppost. A position of a lamppost is given by a pair of integer numbers, X and Y, $-10^8 \leq X, Y \leq 10^8$, corresponding to the plane coordinates.

Output

For each test case in the input your program must produce one single line containing the number of different possible locations for the park.

Examples

Input

```
4
1 7
4 3
3 4
9 1
5
1 7
5 5
2 2
8 8
6 -1
8
1 1
2 2
-2 200
100 3
-6 -6
-51 19
-3 -1
8 -2
```

Output

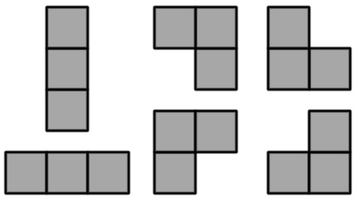
```
3
8
19
```

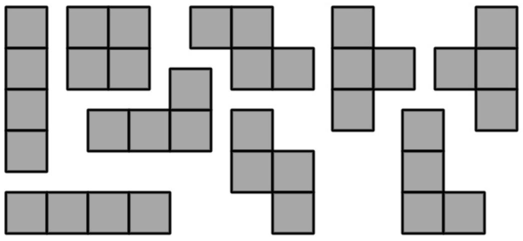
Problem I
Ominobox

Skyrk's planet will never know peace while evil Mago is lurking out there. This time, Mago's malicious scheme was to set up a bomb in the middle of the planet's biggest city. Mago relishes seeing chaos, so instead of detonating the bomb right away, he put a timer on it and left it together with a puzzle. The bomb has a keypad, and the solution to the puzzle disarms the bomb.

The puzzle is called Ominobox; it consists of both a rectangular box with some unit cubes inside and a collection of all possible N-ominoes. Skyrk has to drop every omino somewhere into the box to score points. The maximum score is the solution to the Ominobox.

An N-omino is a collection of N unit squares arranged with coincident sides. A 1-omino is a unit square, and an N-omino is an $(N-1)$-omino with at least one of its sides joined to a unit square.


The six possible 3-ominos


Some of the 19 possible 4-ominos

The box has a rectangular surface and vertical walls; each of the squares of a Cartesian coordinate grid system placed on the box's surface has a non-negative pile of unit cubes. The cubes cannot be moved.

Skyrk will align every omino with the grid squares, and drop it into the box. The omino will fall until it touches a cube or the bottom. Skyrk is not allowed to reflect or rotate the omino and it must lie completely inside the boundaries of the box. The number of points earned with a drop is the distance between the omino and the top of the box. After the drop, Skyrk writes down the points, removes the omino, and drops the next one. The final score is the sum of all points.

The clock is ticking and the countdown on the bomb says 5:00 (five hours!). Can you find out the maximum score Skyrk can get, so that he can disarm the bomb and save the planet's fate from the hands of vile Mago?

Input

The first line in the input contains T ($T \leq 400$) — the number of test cases, after this line T test cases follows. Each test case starts with a line with four integers R, C, H, and N ($1 \leq R, C, H \leq 30$; $1 \leq N \leq 10$) — the box surface dimensions are $R \times C$, the height is H, and the ominoes' order is N. Each of the next R lines contains C integers $H_{i,j}$ ($0 \leq H_{i,j} \leq H$) — the number of cubes at grid square (i, j).

Output

For each test case, print a line containing X, where X is the solution to the Ominobox.

Examples

Input

```
4
2 2 3 1
1 2
0 3
2 2 3 2
1 2
0 3
2 2 3 3
1 2
0 3
2 3 5 4
1 2 5
0 3 4
```

Output

```
3
3
1
5
```

Notes

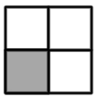


Fig. 1

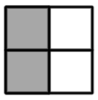


Fig. 2

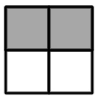


Fig. 3

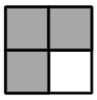


Fig. 4

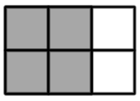


Fig. 5

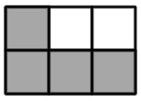


Fig. 6

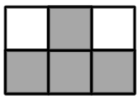


Fig. 7

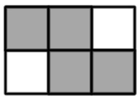


Fig. 8

In the first test case, Fig. 1 shows the best placement of the only 1-omino. The omino hits the bottom of the box at position (1,0) and has a distance of 3 to the top of the box. This placement yields a total of 3 points.

In the second test case, Fig. 2 and Fig. 3 show the best placement of the two 2-ominoes. In Fig. 2 the omino hits a pile of cubes of height 1 at position (0,0) and has a distance of 2 to the top of the box. In Fig. 3 the omino hits a pile of cubes of height 2 at position (0,1) and has a distance of 1 to the top of the box. These placements yield a total of 3 points.

In the third test case, Fig. 4 shows the best placement of the only 3-omino that can fit inside the box. This placement yields a total of 1 point.

In the fourth test case, Fig. 5-8 show the best placement of the four 4-ominoes that can fit inside the box. These placements yield a total of 5 points.

Problem J
Strategy Game

A strategy game with J players is played around a table. Players are identified by numbers from 1 to J and will play a total of R rounds.

At each round each player will play once, in the order of their identifiers; that is, player 1 will play first, player 2 will play second, and so on. Once player J plays, the round is complete, and a next round starts.

A player earns a certain amount of Victory Points every time she or he plays. After all rounds are finished the total points of each player is computed as the sum of Victory Points the player earned on each round. The winner is the player with the maximum number of points; in case of a tie the winner is the player who earned the maximum number of points and played last.

Given the number of players, the number of rounds and a list describing the Victory Points in the order they were obtained, you must determine which player is the winner.

Input

The input contains several test cases. In each test case, the first line contains two integers J and R, respectively the number of players and the number turns ($1 \leq J, R \leq 500$). The second line contains $J \times R$ integers, representing the Victory Points earned by each player in each turn, in the order they happened. The Victory Points obtained in each turn will be always integer numbers between 0 and 100, inclusive.

Output

For each test case in the input, your program must produce one single line, containing the integer representing the winner.

Examples

Input	**Output**

`3 3`
`1 1 1 1 2 2 2 3 3`
`2 3`
`0 0 1 0 2 0`
	`3`
	`1`

Problem K
Palindrome

A palindrome is a string that if reversed is equal to the original string. In other words, it is a string that, when read from back to front, is the same as the original string. For example, BANANAB is a palindrome, while BANANAS is not. In this problem we are interested in a more interesting question.

Given a string S, we want to find a subsequence that is a palindrome. A subsequence is a string that can be obtained from removing zero or more characters from the original string. For example ANNA is a subsequence of BANANAS.

A set of positions of the string S, named special positions, will also be provided. Your task is to find the size of the subsequence that is a palindrome and that includes the largest possible number of special positions. In case there is more than one subsequence that maximizes the number of special positions, you must output the size of the largest subsequence.

Input

The input consists of several test cases. The first line of a test case contains a string S of capital letters with at least one and at most 2000 letters. The second line will contain an integer number N, $0 \leq N \leq |S|$, indicating the number of positions that we are interested to include in the palindrome, followed by N different numbers, between 1 and $|S|$, inclusive, describing the special positions of S.

Output

For each test case from the input your program should print one unique integer number, representing the size of the biggest possible palindrome, as defined above.

Examples

Input	**Output**

```
BANANAS
0
BANANAS
1 7
ACDAAACX
3 2 3 8
MARATONA
4 3 1 5 2
```
	```
	5
	1
	3
	3
	```

Problem L
Lottery

The lottery BWS is played annually. In this lottery N people bet choosing K numbers each. In a formal way, we can say that B_{ij} is the j-th value bet by the i-th person. Then the organizers choose K positive integers. The chosen numbers are called $W_1, W_2, \ldots, W_K$.

The winners are calculated as followed:

- A non-empty subset is chosen randomly from the N participants; in other words, some participants are luckily chosen.
- For each person in this subset the value S_1 is calculated, the sum of all the first numbers bet by them, that is, the sum of the B_{i1} where i is the index of each chosen person. In the same way the values $S_2, \ldots, S_K$ are calculated.
- A parity test between W_j and S_j is performed; in other words, it is verified if the parity (if a number is pair or odd) matches between W_1 and S_1, W_2 and S_2, and so on until W_K and S_K.
- If all parities match, then the people in this subset are considered the winners!

The organizers want to know: is it possible to pick the numbers $W_1, W_2, \ldots, W_K$ so that there is **no** subset of winning participants?

Input

The input contains several test cases. The first line of a test case contains the numbers N ($1 \leq N \leq 30000$) and K ($3 \leq K \leq 50$), which represent the number of participants and the quantity of numbers bet by each person, respectively. The participants bet with integer numbers between 1 and 10^9, inclusive. Each of the next N lines contains K numbers representing the bet of each person, one person per line.

Output

For each test case in the input you must output a single line, containing one letter: '`S`' in case it is possible or '`N`' otherwise.

Examples

Input	Output

```
2 3
1 2 3
5 6 7
3 3
3 2 1
6 5 4
4 4 4
4 3
9 4 7
4 4 4
2 7 2
2 2 1
```
	```
	S
	S
	N
	```

GRAN PREMIO FASE III

Concursos Locales Caribeños 2015 del ACM-ICPC
Gran Premio de México & Centroamérica Fase III
Final Venezolana 2015 del ACM-ICPC (Concurso Nacional)

Conjunto de Problemas del Concurso Real

Documento compuesto por 13 páginas (incluyendo esta portada)

Autores de los problemas y colaboradores:

Carlos Joa Fong (INTEC, República Dominicana)
Jorge Enrique Moreira Broche (Villa Clara, Cuba)
Yaniel Alfredo Velázquez Bruceta (UCI, Cuba)
Yonny Mondelo Hernández (UCI, Cuba)
Nelson González Peñate (UCI, Cuba)
Luis Manuel Diaz Barón (UPR, Cuba)
Alfredo Fundora Rolo (UMCC, Cuba)
Rainel Estrada Montejo (UMCC, Cuba)
Frank Rafael Arteaga Salgado (PSN-ULT, Cuba)
Alberto Eliseo Pacheco Allende (XETID-UCI, Cuba)
Óscar Dávalos Orozco (UP-B, México)

Septiembre 26, 2015.

Problema A – Las Frases de Naebbirac

Descripción

Naebbirac está emocionado por el concurso de este fin de semana. Él quiere llenar todo el sitio con carteles y frases de acuerdo al evento. Por esta razón Naebbirac contrató a una empresa para la tarea; él les dio todas las frases que necesita y ellos hacen frente al proceso de llenar las paredes del sitio con las frases como un Graffiti.

El problema es que el pintor que fue enviado tiene un raro desorden mental llamado Incurable Cambio de Posiciones y Caracteres (ICPC); es decir, que pudiera cambiar un carácter de la frase por otro o incluso el mismo carácter, o puede intercambiar dos caracteres en la frase de sus posiciones respectivas. Después de eso, las frases finales a veces ni siquiera se parecen a lo que se quiere; por ejemplo, para "Bienvenidos_Concursantes" la frase final podría ser "Estamos_muy_bien_aqui_:)". ¡Qué cosa tan loca no cree, pero al menos podemos saber con seguridad que la frase final tendrá el mismo número de caracteres que la frase original! Ahora Naebbirac quiere saber cuántas posiciones tienen diferentes caracteres para cada frase con el fin de corregirlos; es decir pares de caracteres distintos que comparten la misma posición.

La tarea para usted es determinar cuántos caracteres hay que corregir antes del inicio del concurso.

Especificaciones de entrada

La primera línea de entrada contiene un número entero 1 <= T <= 100 que representa la cantidad de casos. Siguen T líneas cada una conteniendo dos cadenas separadas por un único espacio en blanco; la frase original y la frase final respectivamente. Las frases no son vacías y usted puede asumir con seguridad que todas las frases dadas están compuestas por a lo sumo 100 caracteres consecutivos sin espacios.

Especificaciones de salida

Por cada caso imprima una línea con un número entero representando la cantidad de caracteres que hay que corregir antes del inicio del concurso.

Ejemplo de entrada

```
3
Caribbean Naebbirac
Welcome_Contestants We_are_fine_here_:)
Bienvenidos_Concursantes Estamos_muy_bien_aqui_:)
```

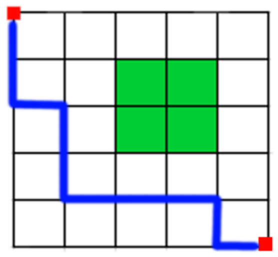
Ejemplo de salida

```
6
17
23
```

Problema B – Cuidado con el Veneno

Descripción

Una hormiga está en una esquina (el punto rojo de la parte superior izquierda en la foto) de una sala de azulejos y quiere llegar a la esquina diagonalmente opuesta (el punto rojo de la parte inferior derecha en la foto). La hormiga sólo puede moverse a lo largo de las líneas entre los azulejos (las fronteras); en cada paso ella siempre reduce la distancia al destino. Pero, ella no puede tocar una pequeña zona cuadrada de 2 x 2 azulejos, donde yace un veneno mortal.



En una habitación de N x N azulejos, las líneas entre las baldosas están convenientemente numeradas entre 0 y N de izquierda a derecha y entre 0 y N de arriba a abajo. Las intersecciones entre dos líneas pueden ser representados como un par de números (a, b); la hormiga debe comenzar en la posición (0, 0) y su destino final debe ser (N, N).

Dada la posición del veneno dentro de la habitación, usted tiene que encontrar el número de rutas que la hormiga puede seguir con seguridad sin tocar las baldosas envenenadas. La hormiga se envenenará si ella toca cualquiera de los cuatro lados de una baldosa envenenada.

Especificaciones de entrada

La entrada consiste en varios casos de prueba, no más de 10^4. Cada caso consta de una primera línea con un número entero N (2 <= N <= 10^6) que representa el tamaño de la habitación, y otra línea que contiene dos números enteros X e Y (0 <= X, Y <= N - 2) separados por un solo espacio en blanco, que representa la posición de la esquina superior izquierda del área cuadrada de 2 x 2 azulejos donde se encuentra el veneno mortal. La última línea de entrada es seguida por una línea que contiene un cero, que no debe ser procesado.

Especificaciones de salida

Para cada caso de prueba usted debe imprimir una línea con un número entero que representa el número de rutas de la hormiga puede seguir de forma segura sin morir porque ha tocado las baldosas envenenadas. Como las respuestas pueden ser muy grandes, usted debe imprimir el resto de dividir la solución por 1000000007 (10^9 + 7).

Ejemplo de entrada

```
5
1 2
4
1 1
0
```

Ejemplo de salida

```
27
2
```

Problema C – Contando Figuras

Descripción

Nelson está enseñando geometría a su hermano pequeño, Willis. Él ha ideado un juego inteligente para evaluar la comprensión de Willis en el tema. El juego es muy simple: Nelson dibuja una cuadrícula y luego Willis tiene que contar el número de figuras de 4, 6 y 8 lados. Aunque es simple, no es obvio: algunas de estas figuras podrían no ser rectángulos o cuadrados.

Nelson dibuja una cuadrícula R x C de acuerdo con las siguientes reglas:

- Bloques vacíos de 1x1 están representados por puntos ('.').
- Bloques rellenos de 1x1 están representados por ceros ('0').
- Los ángulos interiores de las figuras sólo pueden ser de 90 y 270 grados.
- Todas las figuras están completamente llenas de ceros ('0'), no hay agujeros dentro de las figuras. Todos los lados de figuras son paralelos a los ejes de coordenadas.
- Si dos lados son de distintas figuras entonces nunca se tocan entre sí; no hay dos figuras que sean adyacentes (horizontalmente, verticalmente o diagonalmente).

Ayuda a Willis a escribir un programa que pueda hacer este conteo para él.

Especificaciones de entrada

La entrada para este problema consiste en un único caso de prueba. La primera línea de entrada contiene dos números enteros R y C (1 <= R, C <= 100) separados por un único espacio en blanco. Cada una de las R líneas siguientes contiene exactamente C caracteres. Cada caracter es un punto "." o un cero "0", indicando un bloque vacío o relleno respectivamente.

Especificaciones de salida

Usted debe imprimir una línea con tres números enteros separados por exactamente un espacio en blanco que representan el número de figuras con 4, 6 y 8 lados respectivamente.

Ejemplo de entrada

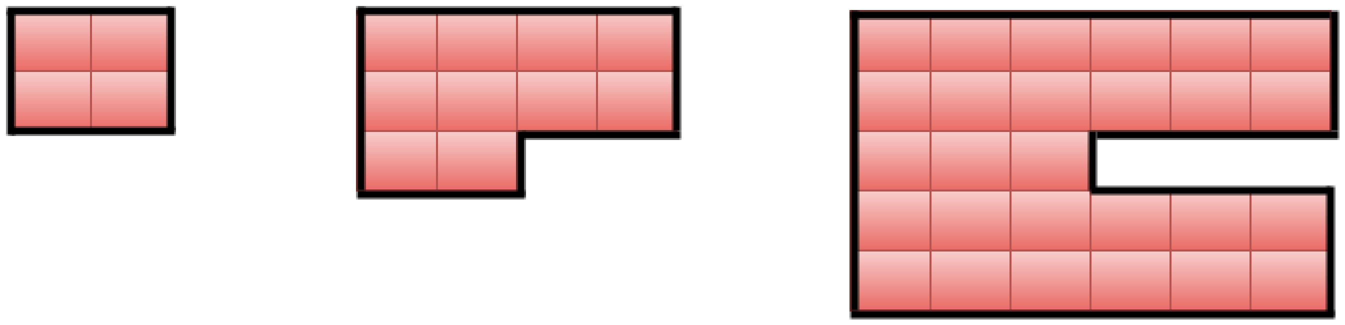
```
10 10
.00.......
.00..0000.
.....0000.
.....00...
..........
.000000...
.000000...
.000......
.000000...
.000000...
```

Ejemplo de salida

```
1 1 1
```

Pistas

En la imagen siguiente se muestran las tres figuras que se encuentran en el caso de ejemplo:



Todos los lados de cada figura han sido dibujados con líneas negras. Usted puede darse cuenta que los lados pueden tener diferentes longitudes. Lo que importa es el número de lados.

Problema D – Dr. B-Tree II

Descripción

Dr. Frank B-Tree está trabajando con grandes cadenas y sus propiedades, más precisamente propiedades palindrómicas.

Una palabra palíndrome es una palabra que se puede leer igual de izquierda a derecha que de derecha a izquierda: "aabcbaa" es palíndrome mientras que "aassab" no, eso también significa que el primer carácter coincide con el último, y el segundo carácter coincide con el anterior al último y así sucesivamente... más precisamente S[i] = S[L - 1 - i] para una cadena S de longitud L indexada con base-cero. Se dice que el índice L - 1 - i es espejo del índice i, y viceversa, por lo tanto un espejo-pareja, (tenga en cuenta que si la palabra tiene una longitud impar el índice L / 2 es un espejo de sí mismo), obviamente en una palabra palíndrome todos los espejo-parejas tienen el mismo valor.

Un casi palíndrome de grado K es una cadena S para la cual exactamente K espejo-parejas tienen el mismo valor. Como un número entero se puede representar como una cadena de caracteres, el Dr. B-Tree quiere encontrar el número de cadenas de longitud L que son casi palíndromes de grado K, y que constan de sólo dígitos. Él también quiere la respuesta modulada por 1000000007 (10^9 + 7).

Especificaciones de entrada

La primera línea de entrada contiene un número entero 1 <= T <= 10^5 que representa la cantidad de casos. Siguen T líneas cada una conteniendo dos números enteros L (1 <= L <= 10^3) y K (1 <= K <= 10^3) separados por un único espacio en blanco; la longitud de las cadenas casi palíndrome y la cantidad de espejo-parejas que deberían tener respectivamente.

Especificaciones de salida

Por cada caso usted debe imprimir una línea con un entero que representa el número de cadenas de longitud L que son casi palíndromes de grado K y que constan de sólo dígitos.

Ejemplo de entrada

```
2
1 1
3 2
```

Ejemplo de salida

```
10
100
```

Problema E – Cantidad Par de Divisores

Descripción

Dados dos números enteros N y M, usted debe encontrar cuántos números enteros k existen tal que N <= k <= M y la cantidad de divisores de k es par.

Especificaciones de entrada

La entrada consiste de múltiples casos de prueba, no más de 1000. Cada caso consiste en una línea con los números enteros N y M (1 <= N <= M <= 10^15) separados por un espacio en blanco. Al último caso de prueba le sigue una línea con dos ceros que no debe ser procesada.

Especificaciones de salida

Por cada caso usted debe imprimir una línea con la cantidad de números enteros entre N y M que tienen una cantidad par de divisores.

Ejemplo de entrada

```
1 2
1 3
2 5
0 0
```

Ejemplo de salida

```
1
2
3
```

Problema F – Juego del Viajero Veloz

Descripción

Viajero Veloz es un juego de mesa en el que los jugadores compiten para llegar primero al final de un camino. Vamos a definir un camino como una secuencia de C cuadrados consecutivos convenientemente numerados entre 1 y C. Los jugadores están convenientemente numerados entre 1 y J y se turnan siempre en el mismo orden. En primer lugar el jugador número 1, segundo el jugador número 2, y así sucesivamente hasta jugador número J. En cada turno un jugador lanza un dado y avanza tantas casillas como el valor del dado indica. Cada jugador comienza con su pieza en la primera casilla del tablero.

Adicionalmente, todos los cuadrados tienen un número entero que representa un movimiento obligatorio que los jugadores deben hacer cuando caigan en esa casilla (sólo después de lanzar un dado). Usted puede asumir con seguridad que el movimiento obligatorio siempre se puede hacer y las piezas nunca van fuera del tablero de juego; es decir, nunca se van antes de la primera o después de la última casilla del tablero. Los movimientos obligatorios deben hacerse de acuerdo con las siguientes reglas:

- Los jugadores deben usar los movimientos obligatorios exactamente una vez por tirada/turno.
- El jugador debe permanecer en la misma casilla del tablero, si su pieza cae en una casilla con el número 0.
- El jugador debe ir hacia atrás si su pieza cae en una casilla con un valor negativo; la pieza se mueve hacia atrás tantas casillas como el movimiento obligatorio indica (valor absoluto del número en la casilla).
- El jugador debe ir hacia adelante si su pieza cae en una casilla con un valor positivo; la pieza se mueve hacia adelante tantas casillas como el movimiento obligatorio indica (valor absoluto del número en la casilla).

El juego llega al final para los jugadores cuando caen en la última casilla del camino; ellos no lanzan más los dados y ganan el juego en el orden en que llegan a esa casilla/cuadrado. Tenga en cuenta que otros jugadores podrían seguir jugando a pesar de ello. Se garantiza que la última casilla siempre tiene un movimiento obligatorio igual a cero. Es su tarea simular el proceso de juego y determinar la secuencia de números que indican los jugadores que ganaron el juego, en ese orden (si es que existe un ganador por lo menos).

Especificaciones de entrada

La primera línea de entrada contiene un número entero T (1 <= T <= 100) que representa el número de casos de prueba. Cada caso de prueba consta de 3 líneas. La primera contiene 3 números enteros: N (2 <= N <= 10) el número de jugadores, S (5 <= S <= 100) el número de casillas del tablero, y D (1 <= D <= 1000) el número de dados lanzados. La siguiente línea contiene los valores de las casillas del tablero y la última línea contiene los valores de los dados lanzados en el orden que se deben tomar para el juego. Las tiradas de los dados se encuentran siempre entre 0 y 9, y se puede asumir con seguridad que los movimientos siempre se pueden hacer y las piezas nunca se van fuera del tablero de juego; es decir, nunca se van después de la última casilla del tablero. Todas las tiradas son válidas y se garantiza que hay al menos un jugador que no está en la última casilla del tablero en ese momento.

Especificaciones de salida

Para cada caso usted debe imprimir una línea con la secuencia de números de los ganadores, en el orden en que llegan al final del camino. Imprima todos los números separados por exactamente un espacio en blanco. En caso de no tener ningún ganador, imprimir el valor -1 en su lugar.

Ejemplo de entrada

```
2
5 8 12
0 -1 5 -1 0 -2 1 0
3 2 7 1 1 5 3 4 1 3 3 3
3 5 8
0 -1 0 -1 0
1 1 1 1 1 1 1 1
```

Ejemplo de salida

```
2 3 1 5 4
-1
```

Problema G – Miedo a la Oscuridad

Descripción

El gato Anders es jefe en su vecindario y tiene como plan expulsar a todos los perros de él, ahora que tiene el control sobre todas las luces. La vecindad consiste en N intersecciones y N - 1 calles conectando todo para de intersecciones. En cada intersección hay un poste de luz e inicialmente todos están encendidos.

Anders puede apagar o encender todas las luces en un camino entre dos intersecciones cualquieras o en solo una de estas y ha decidido mantenerse toda la noche en esta tarea. El perro Alfred quiere visitar algunas intersecciones pero no quiere andar un camino que esté completamente apagado. Por tanto él desea saber cuántas luces hay encendidas en un camino cualquiera entre dos intersecciones.

Su tarea es ayudar a Alfred el perro.

Especificaciones de entrada

En la primera línea un entero N (1 <= N <= 10^5) indicando el número de intersecciones en el vecindario. Las siguientes N - 1 líneas contienen 2 enteros cada una: A y B, cuyos valores están en el rango de 1 a N, indicando que existe una calle entre la intersección A y la intersección B. La siguiente línea contiene un entero M (1 <= M <= 10^4) indicando la cantidad de preguntas a hacerse. Cada una de estas preguntas tiene el siguiente formato:

- 1 A B: Significa que todas las luces en el camino de A hasta B deben ser apagadas, si están encendidas, o viceversa.
- 2 A B: Significa que debe imprimir la cantidad de luces encendidas en el camino que va de A hasta B.

Especificaciones de salida

Por cada pregunta del tipo 2 debe imprimir el número de luces encendidas en el camino indicado.

Ejemplo de entrada

```
6
1 5
2 6
5 4
3 1
5 2
4
1 6 5
2 4 1
1 4 3
2 6 5
```

Ejemplo de salida

```
2
1
```

Problema H – Cantidad de Números

Descripción

¿Cuántos números enteros (incluyendo los enteros negativos) pueden ser escritos, en su representación decimal, con exactamente E dígitos pares y O dígitos impares?

Especificaciones de entrada

La entrada consiste en varios casos de prueba, no más de 200. Cada caso consiste en una línea con un par de números enteros E y O separados exactamente por un espacio en blanco. Estos valores representan el número de dígitos decimales pares e impares, respectivamente, que los números que debe escribir deben tener. E y O cumplen las condiciones E >= 0, O >= 0, y 0 < E + O <= 18. La última línea de entrada es seguida por una línea que contiene dos ceros, que no deben ser procesados.

Especificaciones de salida

Para cada caso de prueba, imprima una línea con un número entero que representa la cantidad de números enteros que cumplen las condiciones anteriores.

Ejemplo de entrada

```
1 1
2 0
0 2
0 0
```

Ejemplo de salida

```
90
40
50
```

Problema I – Lista de Números Naturales

Descripción

Un niño pequeño juega el siguiente juego en su ordenador escribiendo números que se muestran en la pantalla. Él comienza a escribir 1 dos veces, después un 2 entre ellos, luego un 3 entre cada par de números consecutivos cuya suma es 3, luego un 4 entre cada par de números consecutivos cuya suma es 4, y así sucesivamente. Este proceso descrito anteriormente se muestra a continuación:

Paso 1: 1 1
Paso 2: 1 2 1
Paso 3: 1 3 2 3 1
Paso 4: 1 4 3 2 3 4 1

Posteriormente, el niño sigue en la ampliación de la lista hasta llegar al paso 10^12, añadiendo respectivamente cada número; en el i-ésimo paso el número i es escrito entre cada par de números consecutivos cuya suma es i.

Dado un entero positivo N usted debe contar cuántas veces aparece el número N en la lista final.

Especificaciones de entrada

La entrada consiste en múltiples casos de prueba, no más de 125. Cada caso consiste en una línea con un número entero N (1 <= N <= 10^12). La última línea de entrada es seguida por una línea que contiene un cero, que no debe ser procesado.

Especificaciones de salida

Por cada caso usted debe imprimir una línea con un entero que representa el número de veces que N aparece en la lista final.

Ejemplo de entrada

```
2
3
0
```

Ejemplo de salida

```
1
2
```

PREMIO FASE II

Premio Fase II México & Centroamérica
The ACM-ICPC in ESCOM-IPN 2015

October 10th, 2015

Problemset

Authors:

Edgar Augusto Santiago Nieves

Ethan Adrian Jiménez Vargas

Yonny Mondelo Hernández

Document composed of 14 pages including this cover.

Problem A. Alcoholic Pilots

Time Limit:	1 second
Stack Limit:	10 MB
Memory Limit:	32 MB

In one of his many trips, Mr. Ed boarded an airplane where the captain talked like… well, like he was completely drunk. “I would like to greet a very special person here, which has been part of our lives for so much time. His wife says from the control tower that she loves you” – the captain said. Of course, Mr. Ed was really scared, how can alcoholic pilots flight with so many people on their hands? But that was not the worst part, our friend noticed that these drunken pilots like to race between them!

By getting close to the captain’s cabin, Mr. Ed could hear another pilot (drunk, as expected) discussing with the captain by radio. Both of them shared information about how fast they were travelling and how far they were to the nearest airport. “If I arrive earlier to the airport, you will owe me a beer” – the captain bragged, then the airplane started to move abruptly.

Of course Mr. Ed survived, if not, he could not tell us this story. But weirdly, you are wondering who won the race between the pilots and their average arrival time, so you asked the velocity and distance to the airport of both planes. Assume that the planes maintained their velocity even when landing.

Input

The input will contain several test cases. The only line of each test case contains 4 space-separated integers v_1, d_1, v_2 and d_2 ($1 \leq v_1, d_1, v_2, d_2 \leq 10^9$): the velocity and distance to the airport of the plane Mr. Ed and the captain were and the velocity and distance to the airport of the plane the captain was competing with. Velocities are expressed in miles per hour and distances in miles.

The last test case is followed by a single line containing 4 zeroes.

Output

Print 2 lines for each test case. In the first one, you should print “`You owe me a beer!`” if the captain won the race or “`No beer for the captain.`” if the other airplane won the race.

You can safely assume there will be no draws in any test case.

In the second line, print “`Avg. arrival time:`” followed by the average arrival time (in hours) of both airplanes expressed as a simplified fraction of the form x/y, being x and y integers. If the fraction has an integer result, print the result of the division. See format below for more details.

Example

Input	Output

`2 4 1 3`
`1 3 2 4`
`4 7 4 9`
`0 0 0 0`
	`Case #1: You owe me a beer!`
	`Avg. arrival time: 5/2`
	`Case #2: No beer for the captain.`
	`Avg. arrival time: 5/2`
	`Case #3: You owe me a beer!`
	`Avg. arrival time: 2`

Problem B. Bargaining

Time Limit: 3 seconds
Stack Limit: 10 MB
Memory Limit: 32 MB

When he was in the old medina of Marrakech, Mr. Ed and his pals visited a famous Berber market. There were several vendors offering all kind of species, cloths, babouches, lamps and tea pots. As you may imagine, Mr. Ed wanted to buy many souvenirs for all his family. He approached to one of the vendors and asked the price for a pair of babouches, then, the vendor asked "How much are you willing to pay?" Mr. Ed understood that he needed to use his business abilities to set a fair price. "I'll not pay more than 30 euros for that" – said Mr. Ed, "Let me make you a deal, you give me 35 euros, I give you back 10 dirhams and the babouches are yours" – replied the vendor.

Mr. Ed accepted the previous deal and later noticed that he had been cheated! Furious, Mr. Ed decided that he would not be cheated again, so he started to bargain with the vendors to gather information about the prices for a pair of babouches. After several minutes, he managed to bargain with n vendors.

The i-th vendor he bargained with gave euros a value of e_i and dirhams a value of d_i. After realizing that, he came up with an inequality of the form: $e_i\, EUR \pm d_i\, MAD > c_i$, meaning that the total value of euros and dirhams Mr. Ed pays must be more than c_i if he expects the vendor to accept; or $e_i\, EUR \pm d_i\, MAD < c_i$, meaning that he only accepts if the total value of euros and dirhams he pays is less than c_i.

After gathering such information Mr. Ed was ready to buy again. He still had E euros and D dirhams in his pocket, but now you're wondering: which was the effective area of prices he could bargain with that money? The effective area of prices is the area of every possible way Mr. Ed could buy the babouches satisfying the n inequalities, assuming any positive real number of euros and dirhams.

Input

The input will contain several test cases. The first line of each test case contains 3 integers E, D and n, representing the euros and dirhams Mr. Ed had and the number of inequalities gathered ($1 \leq E, D \leq 1{,}000$ and $0 \leq n \leq 1{,}000$). The next n lines contains an inequality as shown in the example, the values for e_i, d_i and c_i will be integers that satisfy $0 \leq e_i, d_i \leq 10{,}000$ and $0 \leq |c_i| \leq 10{,}000$.

The last test case is followed by a single line containing 3 zeroes.

Output

For each test case print a real number with exactly 2 digits after the decimal point, representing the effective area of prices that Mr. Ed could bargain with the vendor (see format below).

Example

Input	Output

`2 2 2`
`1EUR + 0MAD > 1`
`1EUR + 0MAD < 1`
`2 2 2`
`1EUR + 1MAD < 2`
`1EUR - 1MAD > 1`
`0 0 0`
	`Case #1: 0.00`
	`Case #2: 0.25`

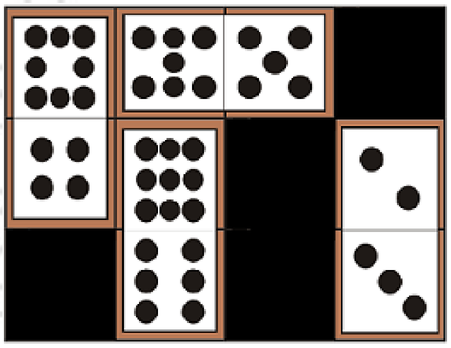
Please note that in this problem there is no relation between Euros (EUR) and Moroccan Dirhams (MAD).

Problem C. Cuban Challenge

Time Limit: 3 seconds
Stack Limit: 10 MB
Memory Limit: 32 MB

Mr. Ed is visiting his pals at Cuba; in particular he wants to have a nice chat with Yonny, who is a very talented domino player. After taking a few rounds of rum, everybody got ready to play domino. As Mr. Ed and Yonny are close friends, they make an invincible team at domino. They won every single game against their friends, so they started to get bored of playing. "Do you feel you play domino as good as a true Cuban?" – asked Yonny. "I play even better" – presumed Mr. Ed.

Now Yonny has challenged Mr. Ed to complete the following task: "I have a wood board divided in n rows of m columns such that there are $n \times m$ squares of equal size on it. The challenge is simple: use as many dominoes as you want to cover each single square in the board without overlapping, but not so fast, that will be pretty easy, right? There are some squares colored in black, you cannot put a domino on this squares. Here, let me show you an example."



"Just because you are my friend, I will let you cut some dominoes in half if you have trouble completing the challenge, but for every domino you cut, you shall pay a small fee to buy more dominoes for me."

As you may expect, now Mr. Ed is all fired up trying to beat Yonny's challenge, so he is trying to answer: which is the minimum number of dominoes he needs to cut in order to fill every single non-black square in the $n \times m$ wood board?

Input

The input will contain several test cases. The first line of each test case contains 2 integers n and m ($1 \leq n \leq 20$ and $1 \leq m \leq 1{,}000$), representing the number of rows and columns in the board. Each of the next n lines contains m characters describing each square in the wood board: '.' means there is an empty square and '#' means there is a black square in the wood board.

The last test case is followed by a single line containing 2 zeroes.

Output

For each test case, print the number of required cuts to complete the Cuban challenge (see format below).

Example

Input	Output
3 4 ...# ..#. #.#. 3 4 ...# ..#. ##.. 0 0	Case #1: 0 Case #2: 1

Problem D. Drinking Game

Time Limit: 3 seconds
Stack Limit: 10 MB
Memory Limit: 32 MB

In the way back home from the World Finals, Mr. Ed and his pals are visiting Madrid, but Mr. Ed is actually very tired and decided to rest a few days, so this problem is not about him.

Ethan and the gang are going to drink some shots in the Gran Via; there, they have a good time drinking and talking about monsters and other terrifying adventures. After several minutes, Ethan came up with a fun drinking game he heard about at the World Finals called "Coprime Shots", he explains:

"Everybody has n piles of empty shot glasses, numbered from 1 to n, the i-th pile has g_i glasses piled up. The objective of the game is to add or remove an arbitrary number of glasses to some of your piles in a way such that, for any pair of distinct piles (i, j) in the set G (result of adding and removing glasses to the original n piles), it holds that $\gcd(G_i, G_j) = 1$. Sounds cool, huh? Anyway, you better play smart, because for every single shot glass you add or remove, you have to fill it up and drink!"

Being a decent programmer, you have no plans on getting wasted tonight, so you decided to minimize the number of shots you have to take in order to win the game.

Input

The input will contain several test cases. The first line of each test contains an integer n: the number of piles in the game ($1 \leq n \leq 100$). The next line contains n integers: the i-th integer represents g_i, the number of shot glasses in the i-th pile ($1 \leq g_i \leq 20$).

The last test case is followed by a single line containing 1 zero.

Output

For each test case, print n positive integers separated by a single space, describing a winning configuration of piles that requires the minimum number of additions and removals to accomplish. Piles should be printed in the same order as in the input. See details in the format below.

If there are multiples solutions to a test, any one of them will be accepted.

Example

Input	Output

3
1 2 3
5
2 4 6 9 10
0
	Case #1: 1 2 3
	Case #2: 1 4 7 9 11

Problem E. Exquisite Strings

Time Limit: 3 seconds
Stack Limit: 10 MB
Memory Limit: 32 MB

Mr. Ed is a very sophisticated man and likes art very much, that is why he and his pals are visiting the amazing museums of Paris. His favourite museum is the Musée d'Chaîne; it has a huge art collection, but the masterpiece of the exhibition is a world famous string of n characters. Mr. Ed has spent hours and hours looking for exquisite pairs of substrings inside the masterpiece.

A substring of a string $S = s_1 s_2 \dots s_n$, represented as $T_{i,j}$ for a pair of indexes $i \le j$, is described as the concatenation $s_i s_{i+1} \dots s_{j-1} s_j$ of characters from string S. Two substrings of S are considered distinct if their indexes i and j are not the same.

The group does not want to observe a single string all day long. In order to leave the museum as soon as possible, you want to help Mr. Ed counting every pair of distinct substrings of the exhibition string that are exquisite. If you don't have as much artistic taste as Mr. Ed, a pair of strings is considered exquisite if they share a common prefix of at least k characters.

If you don't have idea what a prefix is *sigh*, we define it as a substring with starting index i equal to 1.

Input

The first line of input contains a positive integer T representing the number of test cases.

The following T lines contain a non-empty string of $1 \le n \le 10^5$ lowercase letters of the English alphabet representing the museum exhibition string, followed by an integer number $1 \le k \le n$; the length of the minimum prefix required in an exquisite pair of strings.

Output

For each test case in the input, print a single line with an integer representing the number of exquisite substring pairs, modulo 1,000,000,007 ($10^9 + 7$). See format below for details.

Example

Input	Output
5 aaaa 3 ababab 4 cdabcdab 5 qwertyuiop 2 abcabcabcabcx 5	Case #1: 3 Case #2: 7 Case #3: 10 Case #4: 120 Case #5: 313

Problem F. File Transmission

Time Limit: 3 seconds
Stack Limit: 10 MB
Memory Limit: 32 MB

As you may know, Mr. Ed is a very important international businessman. He owns many data centers all around the world, where he keep critical files for his international operations. More specifically, Mr. Ed owns n data centers, numbered from 1 to n. Among these data centers, there are network connections that allow both endpoints of the connection to transfer files with a bandwidth of 50 GB. To ensure complete access to his files, the data centers are connected in such a way that, for any pair (a, b) of distinct data centers, there is at least one way of reaching b from a.

This global network is so powerful that transferring a file along any connection takes no time. Anyway, Mr. Ed knows that the transmission time is no problem in his network, the real problem occurs when he tries to move a file that exceeds the bandwidth of a connection. Explicitly, Mr. Ed cannot transfer instantly a file of 100 GB along a single connection in the network; he will need to transfer one half of the file in one instant and the other half in another instant of time.

Fortunately, data centers can segment a file in multiple parts in order to distribute the transfer load among multiple paths. That way, one file of 100 GB may be transferred instantly. Notice that there might be no way of segmenting a file to achieve this all the time.

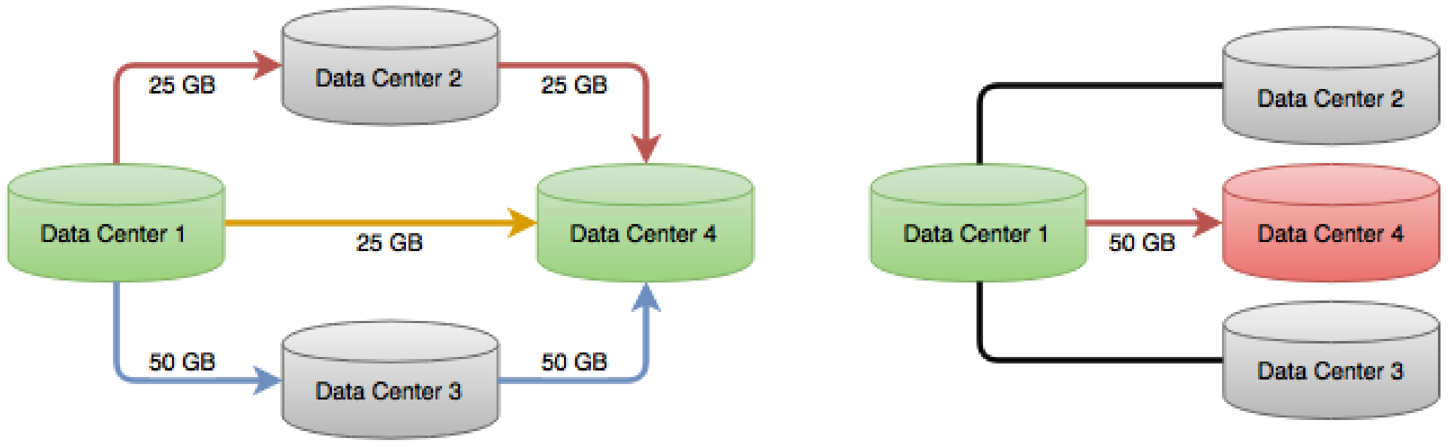


Figure 1. One way of segmenting a file to transfer it from data center 1 to 4 instantly (left).
There is no way to instantly transfer a file of 100 GB from 1 to 4 (right).

To ensure that always is possible to instantly transfer a file of 100 GB between data centers, Mr. Ed can increase (by 1 GB) the bandwidth of any connection by paying 1 dollar.

Mr. Ed do not like to pay for more than he requires, so he asked you to write a software that, given the description of his n data centers and the network connections between them, answers several queries of the type: which is the minimum number of dollars I need to pay in order to assure that I can instantly transfer a file of 100 GB from data center u to data center v?

Input

The input will contain several test cases. The first line of each test case contains 2 integers n and m ($1 \leq n \leq 10{,}000$ and $1 \leq m \leq 20{,}000$): the number of data centers and the number of connections.

The next m lines contains 2 integers a and b each ($1 \leq a, b \leq n$), representing that there is a network connection between data center a and data center b. The will be no connections from any data center to itself and there will be at most one connection between any two data centers.

The next line contains 1 integer q ($1 \leq q \leq 20{,}000$): the number of queries you need to answer. Following this, there will be q lines with 2 integers u and v each ($1 \leq u, v \leq n$), representing the source data center and target data center for the q queries. Test cases end with a blank line.

The last test case is followed by a single line containing 2 zeroes.

Output

For each test case, print the case number followed by q lines with the answer to every query on the test case. Print a single blank line between cases (see format below).

Example

Input

```
4 5
1 2
2 4
1 3
3 4
1 4
2
1 4
3 2

4 3
1 2
1 3
1 4
2
1 4
3 2

0 0
```

Output

```
Case #1:
0
0

Case #2:
50
100
```

The example test cases show the networks illustrated in figure 1. Test case number 1 is the network shown in the left and the second shows the example network in the right.

Problem G. Greedy Artisan

Time Limit: 1 second
Stack Limit: 10 MB
Memory Limit: 32 MB

On their way to the next World Finals, Mr. Ed and his pals are visiting the beautiful city of Moscow. One of their favorite tourism activities is buying souvenirs to bring back home, so they are looking for matryoshkas in a big artisan market close to the Red Square.

In the market, there is a very greedy and clever artisan that sells custom sets of matryoshkas. This artisan has n different matryoshkas in stock, each one having a unique identifier i ($1 \leq i \leq n$), a size s_i and a base price p_i. As the artisan is really clever, he offers a special deal to his clients:

Assume someone wants to buy the custom set $T = \{i_1, i_2, \ldots, i_m\}$ of m matryoshkas. Let us call i_{max} to the identifier of the matryoshka with the maximum size and, in case there are multiple matryoshkas with maximum size, the maximum price in T, then the price one has to pay to buy T is

$$price(T) = \sum_{j=1}^{m} \frac{s_{i_j}}{s_{i_{max}}} \times p_{i_{max}}$$

Mr. Ed wants to exploit the artisan's deal buying exactly k matryoshkas, regardless which are the sizes of each matryoshka. Please determine the minimum number of money he needs to expend.

Input

The input will contain several test cases. The first line of each test case contains 2 space-separated integers n and k, representing the number of matryoshkas the artisan has in stock and the number of matryoshkas Mr. Ed wants to buy ($1 \leq n \leq 100{,}000$ and $1 \leq k \leq n$).

There will follow n lines. The i-th line contains 2 integers s_i and p_i, representing the size and the base price of the i-th matryoshka ($1 \leq s_i, p_i \leq 10^6$). There may be matryoshkas with the same s_i and p_i.

The last test case is followed by a single line containing 2 zeroes.

Output

For each case, print a single line with a real number with 6 digits after the decimal point representing the minimum price Mr. Ed has to pay to buy k matryoshkas (see format below).

Example

Input	Output

3 2
10 5
4 4
6 3
0 0
	Case #1: 5.000000

Problem H. Heavy Luggage

Time Limit:	9 seconds
Stack Limit:	10 MB
Memory Limit:	32 MB

Travelling around the world is really tiring, especially for Mr. Ed and his pals, who carry heavy luggage.

In order to reduce the fatigue, they planned to share the weight of everyone's luggage between their friends. Let's say that person i was carrying w_i kilograms of luggage and this person has f_i friends in the group, then he distributed equitably that weight such that every friend received exactly w_i/f_i kilograms from him. Nobody distributed luggage they had just received.

At the first day of a trip, they distributed the luggage everyone brought from home; by the second day they distributed the luggage received on day one distribution; by the third day of the trip, received luggage from day two is shared; and so on. They kept doing this while they were travelling.

When Mr. Ed arrived home from his latest trip to the World Finals, he noticed that the group forgot to return everyone's luggage! He remembers that there was n people (numbered from 1 to n) travelling in the group, the trip lasted k days and everyone brought a real non-negative number of kilograms at the beginning of the trip. After calling everyone by phone, Mr. Ed wrote down the list of everybody's friends and how many kilograms of luggage they ended up with, including his.

Mr. Ed is exhausted from the trip, so he asked you to find how many luggage each one initially brought.

Input

The input will contain several test cases. The first line of every test case will contain 3 integers n, m and k: the number of people in the group, the number of friendship relations and the number of days the trip lasted ($2 \le n \le 16$, $n \le m \le n \times (n-1)$ and $1 \le k \le 64$).

Each of the next n lines contains a single real number $0 \le w_i \le 1600$ (with up to 200 digits to the right of the decimal point): the kilograms of luggage the i-th person ended up with.

The next m lines contain 2 integers a and b ($1 \le a, b \le n$ and $a \ne b$), each line describing a friendship relation such that person a considers person b a friend. Notice that relations may not be mutual. There will not be repeated relations and every person will consider at least one friend.

The last test case is followed by a single line containing 3 zeroes.

Output

For each test case print n numbers; the i-th number represents the kilograms of luggage person i brought initially to the trip, rounded (half up) to the nearest integer value. You can safely assume that there is at least one solution for each test case, but if there are multiple solutions you must print "`Lost luggage!`" See example below for details about output format.

Example

Input	Output

2 2 7
1.00
1.00
1 2
2 1
3 3 2
1.00
2.00
3.00
1 2
2 3
3 1
3 4 1
1.50
2.00
1.50
1 2
2 1
2 3
3 2
0 0 0
	Case #1: 1 1
	Case #2: 3 1 2
	Case #3: Lost luggage!

First test case is pretty straightforward; both people in the group are mutual friends and they alternated their luggage for 7 days, ending up with 1 kg of luggage each.

For the second test: initially person 1 brought 3 kg of luggage, person 2 brought 1 kg and person 3 brought 2 kg. Person 1 considers person 2 a friend, while person 2 considers person 3 a friend and this last one considers person 1 a friend. After one day of the trip, person 1 gives his initial 3 kg to person 2, this one gives 1 kg of luggage to person 3 and similarly he gives 2 kg to person 1. By the second day, person 1 gives the 2 kg he received in the previous day to person 2, this one gives last day 3 kg to person 3 and finally person 3 passed his 1 kg of luggage to person number 1. This is the only way person 1, 2 and 3 could end up with 1, 2 and 3 kilograms respectively.

There are multiple ways third case result could be achieved, one of them being: person 1 brought 2 kg of luggage, person 2 brought 3 kg and person 3 didn't bring any luggage to the trip.

Problem I. Incredulous Ed

Time Limit:	1 second
Stack Limit:	10 MB
Memory Limit:	32 MB

Today Mr. Ed lent his phone to Ethan. He didn't lock his phone with password because he thought Ethan is a reliable guy, but he wrote "I'm gay" on Ed's timeline. Obviously Mr. Ed is very pissed.

Due to the recent betrayal to his trust, Mr. Ed wants to lock his phone with a pattern. You know what a phone unlock pattern is, right? The lock screen of a phone consists in a grid of n rows and m columns of dots (or circles). Mr. Ed can choose any sequence of dots $S = d_1, d_2, \dots, d_k$ as the phone unlock pattern; this means that, in order to unlock the phone, Ed needs to trace a path starting in d_1 and ending in d_k that passes along every dot in S in exactly the same order.

There are two restrictions to the sequence of dots S: first, every dot d_i must appear at most one time in the sequence and; for every two consecutive dots in S, let's call them d_i and d_{i+1}, the straight line from dot d_i to d_{i+1} must not pass by any dot not previously visited in the sequence. For example, in a 1×3 grid, the sequence $S = 1,3,2$ is not possible since the straight line from 1 to 3 passes through 2 (not visited yet).

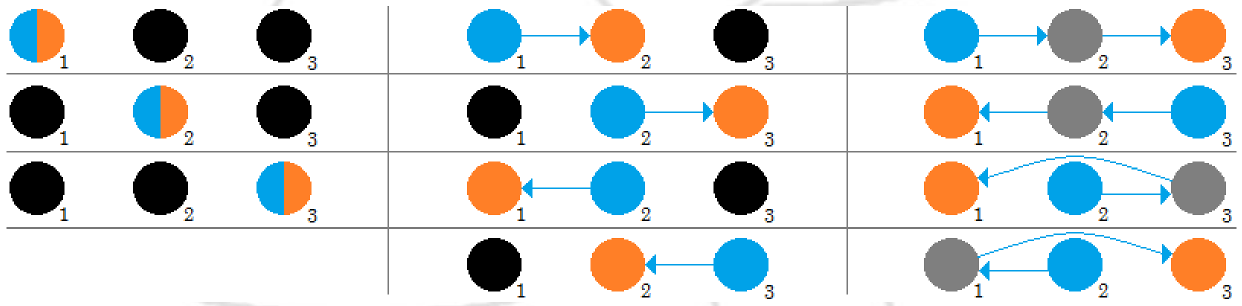


Figure 2. Possible unlock patterns for a 1×3 grid using 1 dot (left), 2 dots (middle) and 3 dots (right).

The above figure illustrates the 11 possible unlock patterns for a lock screen of 1×3 dot grid; blue dots represent the starting dot of the pattern and orange dots represent the respective end dot.

Please help Mr. Ed counting the number of unlock patterns he could use to lock his phone.

Input

The input will contain several test cases. Each test case consists of a single line with two integer numbers $1 \le n \le 3$ and $1 \le m \le 3$: the number of rows and columns in the lock screen grid. The last test case is followed by a single line containing 2 zeroes, which should not be processed.

Output

For each test case, print the number of possible unlock patterns Mr. Ed could use (see format below).

Example

Input	Output

`2 1`
`1 3`
`0 0`
	`Case #1: 4`
	`Case #2: 11`

Problem J. Jair vs Chadan

Time Limit: 3 second
Stack Limit: 10 MB
Memory Limit: 32 MB

Jair and Chadan are very close friends of Mr. Ed; this legendary couple even accompanied him to the World Finals many years ago! When they got bored of fixing windows, these buddies invited Mr. Ed to have dinner tonight and play their favourite hipster board game: "Not a board game".

"Not a board game" is a well-known game among hipsters (probably you never hear about it). In this game, n cards with the numbers from 1 to n are put on a table facing down; each number appears exactly in one card. The objective of the game is to form an integer in base $n + 1$ using all the cards… there is no winner, no score, no rules… you are free to enjoy the game just as it is. Anyway, Chadan is even more hipster than that, he invented an underground way of playing.

Initially k cards are flipped up, then, two players alternate turns to play. On each turn, a player must choose a card and add it to the end of the formed integer. There is only one rule to choose a card: if there is at least one card facing up, the player must choose a face up card; in other case, the player may choose any card he wants. Notice that the player in turn can look the value of each card in the table, even if the card is facing down. After choosing a card and adding it to the formed integer, all the cards with a prime factor of the chosen card are flipped; cards facing up are flipped down and vice versa.

Mr. Ed is watching Jair playing against Chadan. He knows that Jair started the game and aims to form the highest number, while Chadan plays in order to form the lowest number, so he is now wondering: assuming each one plays optimally, which will be the resulting number? By "optimally" we mean that, on each turn, they both choose a card that guarantees no other choice results in a higher (for Jair) or lower (for Chadan) integer at the end of the game.

Input

The first line contains an integer number T corresponding to the number of cases.

The next T lines contains two space-separated integer numbers $1 \le n \le 10{,}000$ and $1 \le k \le \min(100, n)$ representing the amount of cards in the game and the number of cards initially flipped up, followed by k distinct integers representing the initial k cards that had been flipped up.

Output

For each test case output a line with the resulting integer in decimal base (see format below). Since this number can be huge, you must print it modulo 1,000,000,007 ($10^9 + 7$).

Example

Input	Output

3
2 0
3 1 2
7 5 1 2 4 6 7
	Case #1: 7
	Case #2: 39
	Case #3: 1894165

Problem K. Keypad Problem

Time Limit:	3 seconds
Stack Limit:	10 MB
Memory Limit:	32 MB

Back in the Moscow artisan market, Mr. Ed bought an antique calculator which he really wanted to show to his pals. When he came back, he noticed that the keypad of the calculator is incredibly strange!

Instead of having the usual keys from 0 to 9, this weird calculator has n different numeric keys and three extra keys with the operators '+' (plus), '–' (minus) and '=' (equals). The i-th key has a number k_i labeled on it, meaning that by pressing this key, the number k_i is showed on the calculator's screen. If there is some other number currently displayed on the screen, it is replaced by k_i.

As you may notice, this is pretty impractical, but Mr. Ed is clever and knows that in order to display a number not labeled in the keypad, he can make some additions and subtractions of the available numbers. For example, suppose Mr. Ed wants to display the number 7 and the available numeric keys of the keypad are 1, 3 and 5. Our friend could press the third key (with label 5), then press the '+' key (to add another number) followed by the second key (adding 3), press the '–' and first key (in order to subtract 1) and finally get the result of the operations by pressing the '=' key, summing up the number 7.

Mr. Ed wants to display the number r in the calculator's screen, but he is not even sure he can achieve that. Please write a program that tells Mr. Ed if he can display number r and, in case it is possible, shows how to add up r by pressing several keys of the keypad.

Input

The input will contain several test cases (up to 100). The first line of each test contains 2 integers n and r: the number of numeric keys in the keypad and the integer Mr. Ed wants to display ($1 \leq n \leq 50$ and $1 \leq |r| \leq 10^{18}$). The next line contains n integers; the i-th integer represents the number k_i labeled in the i-th numeric key ($1 \leq k_i \leq 20{,}000$). There will be no two keys with the same k_i.

The last test case is followed by a single line containing 2 zeroes.

Output

For each test case, if it is impossible to display number r print "`Stupid keypad!`" If Mr. Ed can display r, print n integers: the i-th integer represents how many times you need to add (if the number is positive) or subtract (if the number is negative) number k_i in order to display r. See format below for details.

Please notice Mr. Ed can press a single key as many times as he like, possibly leading to multiple correct solutions. Any correct solution you print will be accepted.

Example

Input	Output

3 7
1 3 5
1 3
6
0 0
	Case #1: -1 1 1
	Case #2: Stupid keypad!