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

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

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": 1 2 3 4 5
1 2 3 ~~4~~ 5
1 2 3 5
~~1~~ 2 3 5
2 3 5
2 3 ~~5~~
2 3
2 ~~3~~
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.

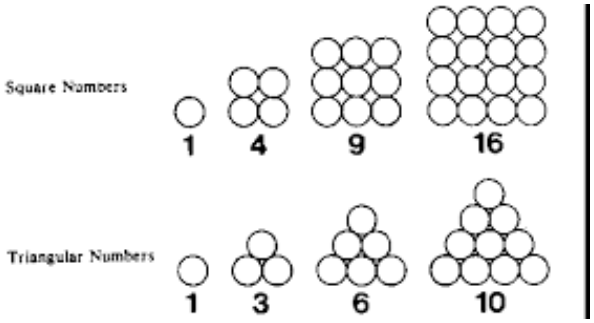
Example

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.

A - Prove Them All 1

B - Baking Cakes with Grandma 2

C - Tennis Championship 3

D - Euler Diagrams 4

E - Going Shopping with Grandma (I) 6

F - Going Shopping with Grandma (II) 8

G - Trading Card Game 9

H - Harvest Moon 11

I - Accelleratii Incredibus 13

J - Ant-Man's Sugar Journey 15

K - Prime Kebab Menu 17

L - The Weakest Link 18

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	Sample Output
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	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.

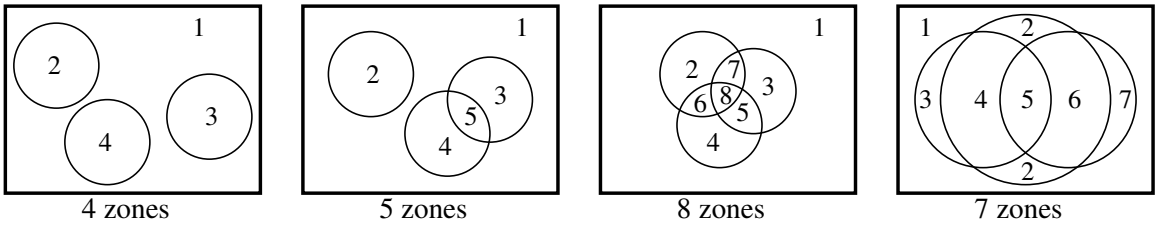
Sample Input	Sample Output
3 2	2 1

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).



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	Sample Output
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	4 5 8 7 5 4 3 2 1 13 6

E - Going Shopping with Grandma (I)

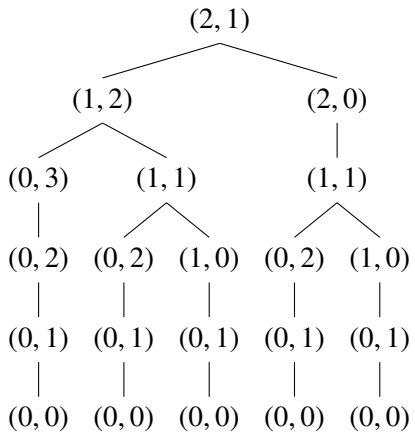
Source file name: `eloi.c`, `eloi.cpp`, *or* `eloi.java`
Author(s): Camilo Rocha

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 6 5 100 2 19 78 1000 1000 0 0	21 31654 5306431377 1942584859 4124225148

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 6 5 100 2 19 78 0 0	1.833333333333 3.166666666667 3.164285714286 13.835714285714 5.207179497838 196.792820502162 7.447739657144 108.552260342856

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 $C = c_1 c_2 \ldots c_n$ 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 $c_i c_{i+1} \ldots c_n c_1 \ldots c_{i-1}$ comes before the string $c_j c_{j+1} \ldots c_n c_1 \ldots c_{j-1}$ 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 `a` < `b` < $\cdots$ < `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