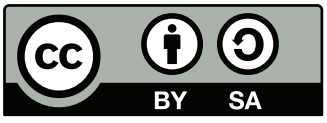
SJAARS KAMPIOENSCHAP PROGRAMMEREN 2014

CONTEST PROBLEM SET

MARCH 24, 2014



A - Administration

Johnny is going to found his own library in a small village. There's only one problem, he needs a system to manage the administration of his new library. Therefore we ask you to help him to create this system. The goal of the system is to read a log and check if it's valid or not. When the log is valid, the system should output the amount each customer needs to pay.

Customers can borrow a book for free, but when a book is not returned within 3 days, a fine is charged of 0,23 € a day. When a customer has not returned a book at the end of the log, he will be charged 10,00 € without the charge per day. Customers can only borrow books that are on a bookshelf. Johnny has one employee, whose job is to bring books back to one of the bookcases. When a customer returns a book to the library it will be placed on a huge pile of books, but a customer can never return a book he didn't borrow. The employee picks up some books from the top of the pile every once in a while. The library will be in small village, therefore he decides to have only one copy of a every book. The log consists of one of the following actions: a user rents a book, a users brings back a certain book or the employee picks up a number of books from the pile.

Input

The input for every test case consists of a single line with the integer $1 \leq n \leq 1000$, the amount of lines of the log. Every next line in the log is one of the following cases:

- When a customer borrows a book: "t - g borrows b"
  Where integer $0 \leq t \leq 10^6$ is the day after the opening of the library, g the name of the user and b the name of the book.

- When a customer returns a book: "t - g returns b"
  Where integer $0 \leq t \leq 10^6$ is the day after the opening of the library, g the name of the user and b the name of the book.

- When an employee returns some books to a bookshelf: "t - n books become available"
  here integer $0 \leq t \leq 10^6 - 1$ is the day after the opening of the library, $1 \leq n \leq 1000$ the number of books that are removed from the pile and become available for borrowing.

Note: every name of a book or a person is a string that doesn't contain any whitespace. Also note that the log is in chronological order.

Output

When the log is not consistent with the rules described above print on one single line the word ”CORRUPT”. Else you must print for every customer the amount they need to pay in the below format. The output lines need to be sorted in alphabetical order on the name of the customers.

Examples

input 1	output 1
6 1 - Sophie borrows Romeo_and_Juliet 2 - Johnny returns Romeo_and_Juliet 3 - Sophie borrows The_Da_Vinci_Code 5 - Berty borrows The_Hunger_Games 5 - 3 books become available 7 - Sophie returns The_Da_Vinci_Code	CORRUPT

input 2	output 2
8 0 - Sophie borrows Romeo_and_Juliet 3 - Sophie borrows The_Da_Vinci_Code 4 - Berty borrows The_Hunger_Games 5 - Sophie returns Romeo_and_Juliet 6 - Berty returns The_Hunger_Games 7 - 1 books become available 7 - John borrows The_Hunger_Games 9 - Sophie returns The_Da_Vinci_Code	Berty E0.00 John E10.00 Sophie E1.15

This page is intentionally left (almost) blank.

B - Back and Forth

Steve has hit the jackpot at his local flea market. He bought a cheap scanner with a special ability: it detects palindrome words! Unfortunately the algorithm that checks the words is broken, thus Steve asked you to write a new algorithm to implement in the scanner and revive it's glory once more.

Remember that a palindrome word is a word that reads the same when reversed: racecar for example is a palindrome.

Input

The input consists of a string s, having length $1 \leq |s| \leq 1000000$.

Output

Your program should output ”beep” if the string s is a palindrome, ”boop” otherwise.

Examples

input 1	output 1
racecar	beep

input 2	output 2
1234564321	boop

input 3	output 3
eeeeeeeeeeeeeeeeeeeeeeeee	beep

C - Cryptography

Dave has just completed the Massive Open Online Course (MOOC) Cryptography on the popular website Coursera.org.
Eager to create his own cryptography system – against the advise of the teacher Dan Boneh to never, ever, ever implement your own crypto-system – he searches for a SKP (Special Key Prime).
A SKP is a prime that is preferably a large number, because the larger the number the more secure it is to use as a key.

Remember that a prime is a number that is only divisible by 1 and itself. For example 2 is a prime because it's only divisible by 1 and 2. 15 however is not a prime since beside 1 and 15, also 3 and 5 happen to divide this number. The number 1 is considered to not be a prime.

Luckily his friend Trudy is quite good at guessing large numbers that could be prime. Your task is given a number by Trudy, to decide whether this is actually a prime or not.

Input

You are given a number $0 \leq n \leq 10^{10}$, the number that Trudy has guessed for Dave to use as a SKP.

Output

You should output "SAFE" (without the quotes) iff the number n is a prime, else your program should output "BROKEN" (again, without the quotes).

Examples

input 1	output 1
2	SAFE

input 2	output 2
15	BROKEN

input 3	output 3
104729	SAFE

D - Diagnosis

In the PHP (Paradise Hospital of Prague) they cure people with a lot of different diseases. The PHP is dealing with a shortage of medical personal, therefore the director is afraid of doctors getting sloppy. Doctors make a diagnosis based on the knowledge they gained in university. So there is a known list of possible diseases and a known list of possible symptoms a patient can have. A doctor makes a diagnosis of one or multiple diseases. Those diseases have known symptoms that need to match exactly with the symptoms of the patient.

More formally: given a set of diseases $D = \{d_1, d_2, ..., d_n\}$, a set of symptoms $S = \{s_1, s_2, ..., s_m\}$ and a function that represents the knowledge about diseases and their symptoms. $f(d) = S'$ with $S' \subseteq S$ and $d \subseteq D$. When a given $D' \subseteq D$ check whether or not the the condition $S = \bigcup_{d \in D'} f(d)$ is true. When it is print "yes" else print "no".

Input

- One line with two integers n and m, with $1 \leq n \leq 1000$ the number of diseases and $1 \leq m \leq 1000$ the number of symptoms. The patient has all symptoms s_1 to s_m.
- One line with one integer k followed by k integers: the set of diseases D' the patient has according to the doctor ($D' \subseteq D$).
- n lines, corresponding to diseases d_1 to d_n, with an integer p followed by p integers: The symptoms belonging to disease i. These symptoms are given in ascending order.

Output

Print "yes" if the symptoms of the diagnosed diseases exactly match the patient's symptoms. Print "no" otherwise.

Examples

input 1	output 1
5 6 3 3 4 5 3 1 2 3 3 2 3 4 2 5 6 3 1 4 5 4 1 2 3 5	yes

input 2	output 2
5 6 4 1 2 4 5 3 1 2 3 3 2 3 4 2 5 6 3 1 4 5 4 1 2 3 5	no

E - Efficient Pinning

While building a computer for Johnny's new library, Eric notices all the shiny pins on the new processor. He also notices, that the processor socket in the motherboard is a lot bigger than usual. After closely examining the pins and holes on the processor and motherboard, he realizes that there are a number of possible pin shapes. It seems the processor can be placed at a number of positions, but only one of them will work.

Unfortunately, Eric has a very busy life, and doesn't have the time to figure out where the processor should go. He will let a friend install the processor. Eric's friend is a very precise worker, but this also makes him slow: it will take him an hour to put in and test a possible position. Every pin on the processor is indicated by either an upercase letter, or $*$ for no pin. Every hole in the motherboard is indicated by an uppercase letter. A pin will fit in the hole if the pin and hole have the same letter, or the pin is $*$. The processor and motherboard have to be placed facing north, as indicated by a big red arrow on both of them, they cant be rotated.

Can you help Eric figure out how many hours he will have to pay his friend to get the computer running?

Input

The first line consists of two space-seperated integers w and h ($1 \leq w, h \leq 400$), indicating the number of columns and rows of the motherboard. After that, h lines of length w follow, each line containing a string of uppercase letters, representing holes in the motherboard socket. On the next line, there are two space-seperated integers n and m ($1 \leq n, m \leq 100$, $n \leq w$, and $m \leq h$), indicating the number of columns and rows of the processor. After that, m lines of length n follow, each line containing a string of uppercase letters and/or $*$, representing pins on the processor.

Output

An integer indicating the number of hours Eric's friend will be working on this computer.

Examples

input 1	output 1
2 2 AB BA 1 2 A B	1

input 2	output 2
3 2 ABA BAA 2 1 *A	3

F - Friends

A new social network has been released and as you're very into new technology, you immediately decide to join it. After creating your account, you would look who has joined the network already so you can send some friend requests. It appears this is not a regular social network like Facebook and there's one difference: when you are friends with someone, you're automatically friends with all the people he or she is also friends with. For example, if you're friends with "Jan", and "Jan" is friends with "Piet", then you are also friends with "Piet" and all other people he has as friend.

You are given the details of the network. You know which members have joined the network and which users are friends with each other. Each user has an unique identifier in the network.

More and more people are joining the new established network and you would like to know whether you're friends with everyone. After all, one can never have too much friends on a social network!

Input

The first line consists of the integers n, m and s ($1 \leq n \leq 1000$, $0 \leq m \leq 25000$, $0 \leq s \leq 1000$): n is the amount of registered users in the network, m denotes the number of connections in the network and s is your user identifier. After that, m lines follow with on each line two integers: a and b which means that users with identifiers a and b are friends in the network.

Output

One line with either yes if you have every other user as a friend or no if not.

Examples

input 1	output 1
5 4 3 0 1 1 2 2 3 3 4	yes

input 2	output 2
6 3 2 1 2 1 4 0 5	no

G - Gardening

Bob has an incredibly huge garden with lots of grass and beautiful flowers, but since he started training his programming skills for the SKP, he does not have that much time to maintain it any more. To reduce the time spent maintaining his garden, Bob selected an area of his garden where he wants to place square stone tiles. He subdivided his garden into a n by n square grid ($1 \leq n \leq 1000$) such that one stone tile fits exactly into one grid cell. Therefore, each tile must be placed inside exactly one grid cell.

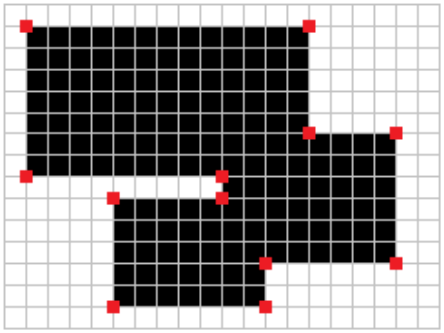


Figure 1 – Example testcase 2, where points given as input are highlighted.

The area Bob wants to fill with tiles is given as a sequence of m points defining its perimeter. Each line segment between points p_i and p_{i+1} defines an edge of the area. Point p_0 is also connected to point p_{m-1}. In each cell within the defined perimeter, exactly one stone tile is placed. Bob now needs your help to count the number of stone tiles he needs to fill the entire designated area.

Input

The first line of the input consists of one integer m ($4 \leq m \leq 1000$): the number of points that define the perimeter of the selected area.

The following input consists of m distinct lines with two space-separated integers x_i and y_i ($1 \leq x_i, y_i \leq 1000$): The coordinates of point p_i are (x_i, y_i). The bottom left corner is defined as point $(0, 0)$ and the top right corner is defined as point (n, n).

Output

One line with the number of square tiles required to fill the entire designated area.

Examples

input 1	output 1
4 6 14 6 33 19 33 19 14	247

input 2	output 2
12 5 1 5 6 10 6 10 7 1 7 1 14 14 14 14 9 18 9 18 3 12 3 12 1	160

H - High Towers

Peter really likes to solve puzzles and his friends know this. They recently asked Peter to solve a well-known puzzle: given a triangular figure of height 3 (see the figure right), how many triangles pointing upwards are in the figure? Peter solved this problem in no-time so he asked his friends for some harder puzzles. And they came up with the same figure of height 4 and height 5 which Peter easily solved.

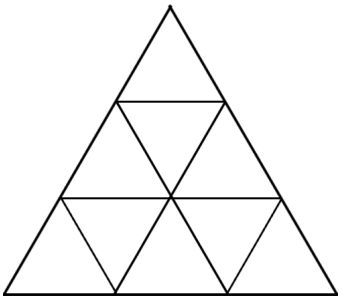


Figure 1 – A triangle puzzle with height 3. There are 10 upward triangles in this figure.

However, Peter got himself in a bit of trouble when his friends want him to solve the puzzle with a figure of height 6. Peter still wants to impress his friends with the correct answer, and he wants to be able to solve the puzzles with an even greater height, possibly up to two million! Since he's a very bad programmer, he asked you for help: given the height of the triangle n, can you determine how many triangles pointing upwards there are visible in the figure?

Input

The input consists of one integer n ($1 \leq n \leq 2 \cdot 10^6$): the height of the triangle.

Output

One integer with the number of triangles pointing upwards in the figure.

Examples

input 1	**output 1**
5	35

input 2	**output 2**
83948	98604181205900

I - Inaccurate Expectations

Jake is doing a lot of programming. Recently Jake has been working on an archiving tool, because organizing isn't one of his strengths. Jake learned that testing is an important part of programming. He needs to make some file generating tool, to make sure his archiving tool is not messing up. Jake has already created a simple generator to generate some folders and files, ready to use his archiving tool on. However the generator seems to take a very long time to finish, if it finishes at all.

It seems Jake has some inaccurate expectations of the effect of his generator. Can you help Jake figure out how many files his generating tool is actually creating, for the given input?

The generator $g(root, n)$ works as follows: in the root folder, n files and n folders are created. In each of those folders, $g(folder, n-1)$ is used to generate the contents of that folder. Calling $g(folder, 0)$ does nothing, of course.

Input

A single integer n, $0 \leq n \leq 1000$

Output

A single integer, the number of files (not the folders) created by $g(root, n)$.

Examples

input 1	output 1
0	0

input 2	output 2
2	4