

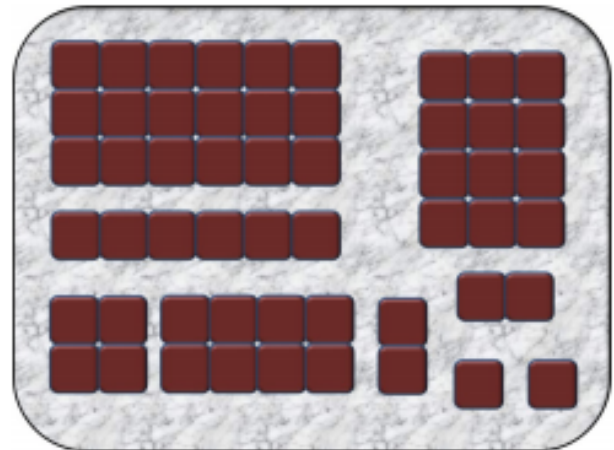
## Problem A. Cutting Brownies




Source file name: cuttingbrownies.c, cuttingbrownies.cpp, cuttingbrownies.java  
Input: Standard  
Output: Standard







John Horton Conway (1937-) is a British mathematician with many contributions to mathematics. He is famous for the invention of the cellular automaton, more popularly known as the “Game of Life.” This problem is inspired by a game Conway invented in the 1970s.











This game is played using a rectangular sheet of brownies fresh out of the oven. The players are Harry Horizontal and Vicky Vertical. Initially, there is a single piece consisting of  $B \times D$  connected squares (the individual brownies).





At each turn, a player chooses one of the remaining pieces and if possible, cuts it into two smaller pieces such that both pieces have integer breadth and depth. Harry may make only horizontal cuts, Vicky only vertical cuts. Pieces may not be rotated before or after a cut. If a player cannot cut any of the remaining pieces, that player loses.



Let’s consider some examples. The simplest game is . In this case, neither Harry nor Vicky can make a move, so whoever starts loses. On the other hand,  is a win for Harry, no matter who starts. Similarly,  is a win for Vicky, no matter who starts.

Consider  , which is a loss for whoever starts. For instance, if Vicky starts, her only move leaves Harry with  ,  and once he cuts any of the pieces, Vicky is left with  ,  ,  (in any order) and thus again without moves. For reasons of symmetry, Harry loses if he is made to start.

Intuition might tell us that Vicky should tend to win if the initial sheet is broader than it is deep (since such sheets yield more opportunities for vertical cuts), but consider  . If Harry starts, his only possible move leaves Vicky with  ,  and a win. But if Vicky starts, any possible move leaves Harry with  ,  . Harry responds and leaves Vicky with  ,  ,  , which Vicky will eventually lose since there are no moves left in the 2  sheets and whoever makes the first move on  loses.

On the other hand,  is a winner for Vicky, no matter who starts. If Harry starts, he runs out of moves after his first cut. If Vicky starts, her best move is to cut in the center, leaving Harry with  ,  , which he loses because each  game is lost by whoever moves first.

Given the initial size of the sheet, and given who starts the game, write a program that computes if the starting player has a strategy to force a win!



## Input

The first line contains an integer  $1 \leq N \leq 10$  denoting the number of test cases that follow. Each test case consists of a single line containing two integers  $B$  and  $D$ , and a string  $S$ . Here  $B$  denotes the initial breadth of the sheet ( $1 \leq B \leq 500$ ),  $D$  denotes the initial depth of the sheet ( $1 \leq D \leq 500$ ) and  $S$  is either Harry or Vicky depending on whether Harry or Vicky moves first.

## Output

For each test case, output whether the player who starts can force a win in the game. Output the player's name followed by `can win` or `cannot win`.

## Example

Input	Output
5	Harry cannot win
1 1 Harry	Vicky cannot win
2 2 Vicky	Vicky cannot win
3 2 Vicky	Vicky can win
4 2 Vicky	Harry can win
6 8 Harry	

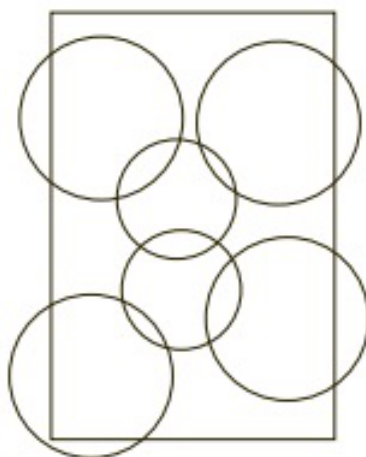
## Problem B. UnDetected

Source file name: undetected.c, undetected.cpp, undetected.java  
 Input: Standard  
 Output: Standard

The Department of Defense has been designing autonomous robots that can infiltrate war zones and other hostile places in order to carry out missions. Now they want to test their latest design, the Penetrator1700, and they've hired you to help design the test environment.

The test environment is a rectangular field with some sensors placed within the field. Each sensor has a certain radius defining the region within which it can detect a robot. You want to design the field to have as many sensors as possible while still permitting a route across the field that avoids detection.

The field is a region of the coordinate plane defined by  $0 \leq x \leq 200$  and  $0 \leq y \leq 300$ . The robot can be modeled by a point that must remain on the field at all times. It starts at the bottom of the field ( $y = 0$ ) and must end at the top of the field ( $y = 300$ ), and must not pass within range of any sensor. There are  $N$  sensor locations given by triples  $(x, y, r)$  of integers, where each  $(x, y)$  is a point on the field, and  $r$  is its radius of detection. The implied sensor circles may overlap, but will never be tangent with each other nor with the boundary of the field. All sensors are initially inactive. You must find the largest value of  $k$  such that if sensors 1, 2, 3,  $\dots$ ,  $k$  are activated there is a path for the robot across the field, but no path if the  $(k + 1)$ st sensor is also activated. It is guaranteed that there is no path if all  $N$  sensors are activated.



Sensor circles corresponding to the first three sample inputs.

### Input

Input begins with a positive integer  $N \leq 200$ . Each of the next  $N$  lines has three space-separated integers, representing  $x, y, r$  for a sensor, where  $r \leq 300$ . All sensors lie at different  $(x, y)$  positions. The first three sample inputs below correspond to the figure shown.

### Output

Output a single integer (which may be 0) giving the largest  $k$  as described above.



## Example

Input	Output
6 36 228 58 164 224 58 88 170 42 93 105 42 167 85 58 28 44 58	2
6 36 228 58 28 44 58 164 224 58 88 170 42 93 105 42 167 85 58	3
6 28 44 58 36 228 58 88 170 42 93 105 42 164 224 58 167 85 58	4
3 100 150 101 30 30 10 170 30 100	0

## Problem C. Disastrous Downtime

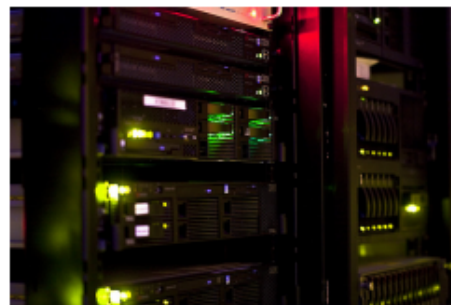
Source file name: `downtime.c`, `downtime.cpp`, `downtime.java`  
 Input: **Standard**  
 Output: **Standard**

You're investigating what happened when one of your computer systems recently broke down. So far you've concluded that the system was overloaded; it looks like it couldn't handle the hailstorm of incoming requests. Since the incident, you have had ample opportunity to add more servers to your system, which would make it capable of handling more concurrent requests. However, you've simply been too lazy to do it—until now. Indeed, you shall add all the necessary servers . . . very soon!

To predict future requests to your system, you've reached out to the customers of your service, asking them for details on how they will use it in the near future. The response has been pretty impressive; your customers have sent you a list of the exact timestamp of every request they will ever make!

You have produced a list of all the  $n$  upcoming requests specified in milliseconds. Whenever a request comes in, it will immediately be sent to one of your servers. A request will take exactly 1000 milliseconds to process, and it must be processed right away.

Each server can work on at most  $k$  requests simultaneously. Given this limitation, can you calculate the minimum number of servers needed to prevent another system breakdown?



Claus Rebber, cc-by-sa

### Input

The first line contains two integers  $1 \leq n \leq 100000$  and  $1 \leq k \leq 100000$ , the number of upcoming requests and the maximum number of requests per second that each server can handle.

Then follow  $n$  lines with one integer  $0 \leq t_i \leq 100000$  each, specifying that the  $i^{th}$  request will happen  $t_i$  milliseconds from the exact moment you notified your customers. The timestamps are sorted in chronological order. It is possible that several requests come in at the same time.

### Output

Output a single integer on a single line: the minimum number of servers required to process all the incoming requests, without another system breakdown.

### Example

Input	Output
2 1 0 1000	1
3 2 1000 1010 1999	2

## Problem D. Entertainment Box

Source file name: entertainmentbox.c, entertainmentbox.cpp, entertainmentbox.java  
Input: Standard  
Output: Standard

Ada, Bertrand and Charles often argue over which TV shows to watch, and to avoid some of their fights they have finally decided to buy a video tape recorder. This fabulous, new device can record  $k$  different TV shows simultaneously, and whenever a show recorded in one the machine's  $k$  slots ends, the machine is immediately ready to record another show in the same slot.

The three friends wonder how many TV shows they can record during one day. They provide you with the TV guide for today's shows, and tell you the number of shows the machine can record simultaneously. How many shows can they record, using their recording machine? Count only shows that are recorded in their entirety.



### Input

The first line of input contains two integers  $n, k$  ( $1 \leq k < n \leq 100000$ ). Then follow  $n$  lines, each containing two integers  $x_i, y_i$ , meaning that show  $i$  starts at time  $x_i$  and finishes by time  $y_i$ . This means that two shows  $i$  and  $j$ , where  $y_i = x_j$ , can be recorded, without conflict, in the same recording slot. You may assume that  $0 \leq x_i < y_i \leq 1000000000$ .

### Output

The output should contain exactly one line with a single integer: the maximum number of full shows from the TV guide that can be recorded with the tape recorder.

### Example

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

## Problem E. Safe Passage

Source file name: safepassage.c, safepassage.cpp, safepassage.java  
Input: Standard  
Output: Standard

A group of friends snuck away from their school campus, but now they must return from the main campus gate to their dorm while remaining undetected by the many teachers who patrol the campus. Fortunately, they have an invisibility cloak, but it is only large enough to cover two people at a time. They will take turns as individuals or pairs traveling across campus under the cloak (and by necessity, returning the cloak to the gate if others remain). Each student has a maximum pace at which he or she is able to travel, yet if a pair of students are walking under the cloak together, they will have to travel at the pace of the slower of the two. Their goal is to have everyone back at the dorm as quickly as possible.

As an example, assume that there are four people in the group, with person A able to make the trip in 1 minute, person B able to travel in 2 minutes, person C able to travel in 7 minutes, and person D able to travel in 10 minutes. It is possible to get everyone to the dorm in 17 minutes with the following plan:

- A and B go from the gate to the dorm together (taking 2 minutes).
- A returns with the cloak to the gate (taking 1 minute).
- C and D go from the gate to the dorm together (taking 10 minutes).
- B returns with the cloak to the gate (taking 2 minutes).
- A and B go from the gate to the dorm together (taking 2 minutes).



Photo by Ian Hunt

### Input

The input is a single line beginning with an integer,  $2 \leq N \leq 15$ . Following that are  $N$  positive integers that respectively represent the minimum time in which each person is able to cross the campus if alone; these times are measured in minutes, with each being at most 5000. (It is a very large campus!)

### Output

Output the minimum possible time it takes to get the entire group from the gate to the dorm.

### Example

Input	Output
2 15 5	15
4 1 2 7 10	17
5 12 1 3 8 6	29



## Problem F. Hero Power

Source file name: heropower.c, heropower.cpp, heropower.java  
 Input: Standard  
 Output: Standard

Rhythm gaming seems to be having a bit of a renaissance this October, with both a new “Rock Band” and a “Guitar Hero” game coming out. Bj0rn is preparing to achieve top scores in “Guitar Hero Live”, but he needs your help in figuring out what the maximum score is for all the new songs. Apparently, the new game has something called Hero Power, but Bj0rn is betting that it really is the same thing as the “Star Power” that has always been in these games.

“Guitar Hero’s” scoring essentially works as follows: the player advances along a note chart and scores one point for each note he hits. Bj0rn will settle for nothing less than perfection; every note will be hit!



However, there’s an added twist: “Star Power!”—simply called *SP*. Every now and then, a streak of star-shaped notes appear on the note chart. These streaks are *SP phrases*. When between the first and last note of an SP phrase, the player has the ability to charge up a so-called *SP meter*, which stores the amount of time the player has spent charging it. You can start charging at the exact moment of the first note and all the way up till the last note. You can also pause charging at any time and you do not have to use the accumulated SP immediately after you stop charging, so it is possible to accumulate SP charge from multiple phrases.

When the SP meter contains a positive amount of seconds, at any point in the song—even at the exact moment of a note—the player is free to activate Star Power. From this moment, the SP meter starts draining until it is completely empty. For example, if it contains  $\pi + \sqrt[4]{7}$  seconds of SP, it will take  $\pi + \sqrt[4]{7}$  seconds to drain completely. During an activation, every note is worth two points as long as the SP meter is non-empty! In particular, if you start activating at the exact moment of a note, that note is already worth two points and if you hit a note during the last moment of activation, that note is only worth one point, because the SP meter has just become empty.

There is a downside to activating Star Power. If an SP activation overlaps with an SP phrase and the SP meter is positive at some point during the overlap, the SP phrase degrades back to plain notes. In particular, if you hit the first note of an SP phrase on the exact moment when the SP meter drains to 0, the SP phrase is not degraded. It’s fine to activate mid-phrase, but the rest of the phrase still suffers from the overlap and disappears, so you can not charge more Star Power from that phrase.

Can you help Bj0rn find the best strategy and figure out how many points he can get?

### Input

The first line of input consists of two integers  $1 \leq n \leq 50000$  and  $0 \leq p \leq 100$ , the number of notes and SP phrases respectively. The second line is a strictly increasing sequence of  $n$  integers  $0 \leq t_i \leq 50000000$ , the positions of all notes in milliseconds. Then follow  $p$  lines containing two integers each,  $0 \leq s_i \leq e_i \leq 50000000$ , the positions of the start and end of the  $i$ ’th Star Power phrase.

Notes are guaranteed to exist on the start and end positions of each SP phrase. SP phrases never overlap and are given in ascending order.

### Output

The maximum score as a single integer.



**Example**

Input	Output
3 1 0 10 20 0 10	4
6 1 0 10 20 26 40 50 0 40	9
10 2 0 10 20 30 40 50 60 70 80 90 0 40 70 80	14

## Problem G. Goblin Garden Guards

Source file name: goblingardenguards.c, goblingardenguards.cpp, goblingardenguards.java  
Input: Standard  
Output: Standard

In an unprecedented turn of events, goblins recently launched an invasion against the Nedewsian city of Mlohkcots. Goblins—small, green critters—love nothing more than to introduce additional entropy into the calm and ordered lives of ordinary people. They fear little, but one of the few things they fear is water.

The goblin invasion has now reached the royal gardens, where the goblins are busy stealing fruit, going for joyrides on the lawnmower and carving the trees into obscene shapes, and King Lrac Fatsug has decreed that this nonsense stop immediately!

Thankfully, the garden is equipped with an automated sprinkler system. Enabling the sprinklers will soak all goblins within range, forcing them to run home and dry themselves.

Serving in the royal garden guards, you have been asked to calculate how many goblins will remain in the royal garden after the sprinklers have been turned on, so that the royal gardeners can plan their next move.



Felipe Escobar Bravo, cc-by-nc-nd

### Input

The input starts with one integer  $1 \leq g \leq 100000$ , the number of goblins in the royal gardens.

Then, for each goblin follows the position of the goblin as two integers,  $0 \leq x_i \leq 10000$  and  $0 \leq y_i \leq 10000$ . The garden is flat, square and all distances are in meters. Due to quantum interference, several goblins can occupy exactly the same spot in the garden.

Then follows one integer  $1 \leq m \leq 20000$ , the number of sprinklers in the garden.

Finally, for each sprinkler follows the location of the sprinkler as two integers  $0 \leq x_i \leq 10000$  and  $0 \leq y_i \leq 10000$ , and the integer radius  $1 \leq r \leq 100$  of the area it covers, meaning that any goblin at a distance of at most  $r$  from the point  $(x_i, y_i)$  will be soaked by this sprinkler. There can be several sprinklers in the same location.

### Output

Output the number of goblins remaining in the garden after the sprinklers have been turned on.

### Example

Input	Output
5 0 0 100 0 0 100 100 100 50 50 1 0 0 50	4

## Problem H. Just a Quiz

Source file name: justaquiz.c, justaquiz.cpp, justaquiz.java  
Input: Standard  
Output: Standard

In the TV quiz *Montermind*, a contestant chooses a topic and is then asked questions about it during a fixed period of time. The contestant earns one point for each correct answer. When the time runs out, the contestant must be silent.

Teresa has figured out such a niche topic that she knows all possible questions that may be asked about it, as well as all the answers. Since the competition is fierce, she has decided to sometimes answer a question before the host finishes reading it. The host picks each question uniformly at random from the pool of possible questions, and each question may be asked multiple times. When reading a question, the host reads at a pace of one word per second.

Teresa can interrupt the host mid-question—between words, or even before hearing the first word—but not mid-word—that would be extremely impolite. Answering also takes one second, and the host will start reading another question immediately after an answer—unless Teresa interrupts again.

She wrote a program to help her choose the best moment to answer, and now there is only one question left for you. How many points does she expect to score?

For example, in the first sample test case the answer is completely determined after hearing one word, so it is optimal to answer after hearing it, and Teresa answers 2 questions correctly in 4 seconds. In the second sample test case, if the first word is *What*, then it takes too much time to wait for the question to finish. Therefore Teresa says *Now!* 4 times and expects to get 1/3 of the answers right.

### Input

The first line contains two integers  $t$  and  $n$  ( $1 \leq t \leq 100, 1 \leq n \leq 100000$ ), the duration of the quiz and the number of questions. Each of the following  $n$  lines contains a question, which is a space-separated list of words terminated by a question mark; and an answer, which is a single word.

Each word is a sequence of non-space ASCII printable characters, between the ASCII values of '!' and '~'. Only the last word of a question has a question mark ('?'). You can assume that no question is a prefix of another and that punctuation marks are part of a word. Words spelled with different upper/lower case are assumed to be different.

It is guaranteed that the total number of word characters is at most 100000.

### Output

Output the expected score of an optimal strategy. Answers within a relative or absolute error of  $10^{-6}$  will be accepted.



Vincent's Stuhl mit Pfeife



## Example

Input
4 4 How much is 6 times 9? 42 How much is 9 times 6? 42 Is there intelligent life on Earth? Probably What is the air speed velocity of an unladen swallow? African?
Output
2.0000000000

Input
4 3 What do we send? Code What do we want? Accepted When do we want it? Now!
Output
1.333333333

## Problem I. Adjoin the Networks

Source file name: `adjoin.c`, `adjoin.cpp`, `adjoin.java`  
Input: **Standard**  
Output: **Standard**

One day your boss explains to you that he has a bunch of computer networks that are currently unreachable from each other, and he asks you, the cable expert's assistant, to adjoin the networks to each other using new cables. Existing cables in the network cannot be touched.

He has asked you to use as few cables as possible, but the length of the cables used does not matter to him, since the cables are optical and the connectors are the expensive parts. Your boss is rather picky on cable usage, so you know that the already existing networks have as few cables as possible.

Due to your humongous knowledge of computer networks, you are of course aware that the latency for an information packet travelling across the network is proportional to the number of *hops* the packet needs, where a hop is a traversal along a single cable. And since you believe a good solution to your boss' problem may earn you that long wanted promotion, you decide to minimise the maximum number of hops needed between any pair of network nodes.



Wikimedia, cc-by-sa

### Input

On the first line, you are given two positive integers, the number  $1 \leq c \leq 10^5$  of computers and the number  $0 \leq l \leq c - 1$  of existing cables. Then follow  $l$  lines, each line consisting of two integers  $a$  and  $b$ , the two computers the cables connect. You may assume that every computer has a unique name between 0 and  $n - 1$ .

### Output

The maximum number of hops in the resulting network.

### Example

Input	Output
6 4 0 1 0 2 3 4 3 5	3
11 9 0 1 0 3 0 4 1 2 5 4 6 4 7 8 7 9 7 10	4

## Problem J. ICar

Source file name: icar.c, icar.cpp, icar.java  
Input: Standard  
Output: Standard

You are at home and about to drive to work. The road you will take is a straight line with no speed limit. There are, however, traffic lights precisely every kilometer, and you can not pass a red light. The lights change instantaneously between green and red, and you can pass a light whenever it is green. You can also pass through a light at the exact moment of changing colour. There are no traffic lights at the start or the end of the road.

Now your car is special; it is an iCar, the first Orange car, and it has only one button. When you hold down the button, the car accelerates at a constant rate of  $1m/s^2$ ; when you release the button the car stops on the spot.

You have driven to work many times, so you happen to know the schedules of the traffic lights. Now the question is, how quickly can you get to work?



Cropped from picture by Les Chatfield

### Input

The first line contains a single integer  $n$ , the length of the road in kilometers ( $1 \leq n \leq 16$ ). Each of the next  $n - 1$  lines contains 3 integers  $t_i$ ,  $g_i$  and  $r_i$ , the first time the  $i$ -th light will switch from red to green after the moment you start driving the car; the green light duration, and the red light duration ( $40 \leq g_i, r_i \leq 50$ ;  $0 \leq t_i < g_i + r_i$ ). Times are given in seconds.

You may assume that any light with  $t_i > r_i$  is green at the time you start driving the car, and switches to red  $t_i - r_i$  seconds later.

### Output

Output the minimum time required to reach the end of the road. Answers within a relative or absolute error of  $10^{-6}$  will be accepted.

### Example

Input	Output
1	44.72135955
2 50 45 45	68.52419365
2 25 45 45	63.2455532

## Problem K. Quick Brown Fox

Source file name: quickbrown.c, quickbrown.cpp, quickbrown.java  
 Input: Standard  
 Output: Standard

A *pangram* is a phrase that includes at least one occurrence of each of the 26 letters, 'a' ... 'z'. You're probably familiar with this one: "The quick brown fox jumps over the lazy dog."

Your job is to recognize pangrams. For phrases that don't contain every letter, report what letters are missing. We'll say that a particular letter occurs in the phrase if it occurs as either upper case or lower case.



Photo by Neil McIntosh

### Input

Input starts with a line containing an integer  $1 \leq N \leq 50$ . The next  $N$  lines are each a single phrase, possibly containing upper and lower case letters, spaces, decimal digits and punctuation characters '.', ',', '?', '!', ' ' and ' " '. Each phrase contains at least one and no more than 100 characters.

### Output

For each input phrase, output "pangram" if it qualifies as a pangram. Otherwise, output the word "missing" followed by a space and then the list of letters that didn't occur in the phrase. The list of missing letters should be reported in lower case and should be sorted alphabetically.

### Example

Input
3 The quick brown fox jumps over the lazy dog. ZYXW, vu TSR Ponm lkj ihgfd CBA. .,?!' " 92384 abcde FGHIJ
Output
pangram missing eq missing klmnopqrstuvwxyz