Problem A. Appalling Architecture

Source file name:	appalling.c, appalling.cpp, appalling.java, appalling.py
Input:	`Standard`
Output:	`Standard`

You have recently been hired as an architect for the BAPC (Bureau of Architecture and Promising Constructions), responsible for top-quality buildings such as the Tower of Pisa. However, in the past couple of weeks, some of the constructions that the BAPC has made have collapsed! It is up to you to figure out whether any other constructions are in danger.

Picture by Carlos ZGZ on Flickr.

After some research it seems like the x-coordinate of the center of gravity of some of the constructions is off: if this is too much to the left or to the right, the construction will fall over. Hence, you decide to check all the blueprints and see whether the constructions are stable or not.

Given is an up to 100 by 100 grid of characters in `.#/\_|-`.

The `.` characters denote empty space, while each other character represents a completely filled 1×1 box (any difference in symbols used is due to the artistic freedom of the other architects), whose center of mass is at the center of the box.

Every construction forms a single connected component that touches the ground, i.e. the bottom layer of the grid.

The construction falls to the left if the x-coordinate of the center of gravity is less than the x-coordinate of the leftmost point of the construction that touches the ground, and it falls to the right if the x-coordinate of the center of gravity is larger than the x-coordinate of the rightmost point of the construction that touches the ground. It is guaranteed that the center of gravity is never exactly above the leftmost or rightmost point where the building touches the ground.

Given a blueprint, is the construction balanced, does it fall to the left, or does it fall to the right?

Input

- The first line has $1 \leq h \leq 100$ and $1 \leq w \leq 100$, the height and width of the grid.
- Then follow h lines with l characters each. Each character is either `.`, indicating empty space, or one of `#/\_|-`, indicating a filled 1×1 box.

Output

- Print a single line containing `left`, `balanced`, or `right`.

Example

Input	Output
3 3 /-\ |.| #.#	balanced
3 3 ... --- ..|	left
3 3 ./\ .\/ .|.	balanced
20 19 ....-____--.. ...._/\\////-. ....-//#\#\/_.. ...._/\///////_. ......-/#######\/-. ......////\/#\#/_.. ....../\#\#/////_.. ....._/#####\#\/... ....._/\////////-.. ...../########/-... ...._///\/#\#\/.... ...._/#/#/\/\/_.... ..._/\#######/-.... ..._/\////////-.... .../\#\#\#\#/-..... .._///\\\\#\/-..... .._//////////-..... ..////##////-...... .-____##____.......	balanced

Problem B. Bee Problem

Source file name:	bee.c, bee.cpp, bee.java, bee.py
Input:	`Standard`
Output:	`Standard`

You are a busy little bee, and you have a problem. After collecting nectar all day long, you are returning to the beehive with a large supply of honey. You would really like to take a nap now, but sadly, you have to store all the honey in your beehive first. Opening up a cell in the hive to funnel honey into takes a lot of time, so you want to avoid doing this as much as possible.

Some of the cells in the beehive are already filled with old, hardened honey. The other cells are still empty. If you pour honey into an empty cell, it will automatically start flowing into adjacent empty cells. From these cells, the honey will again flow to other neighbouring empty cells. This saves you from having to funnel honey into them directly. You decide to use this to your advantage, by pouring into cells with lots of (indirect) adjacent open cells.

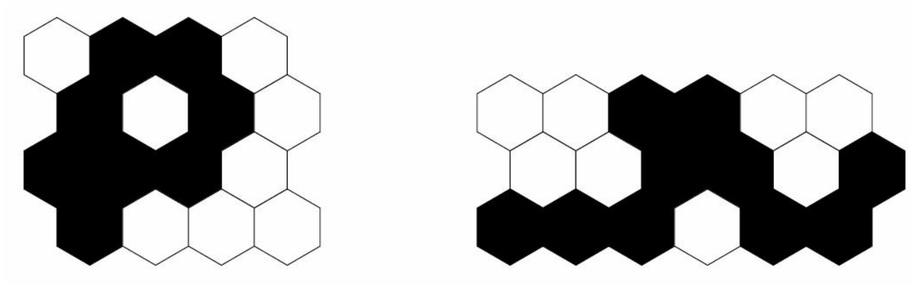


Figure 1: The beehives from the first two samples. The black hexagons already contain hardened honey. The white cells are still empty. You have some units of honey, and know the layout of your beehive. By cleverly choosing which cells to funnel honey into, what is the minimal amount of work you have to do?

Input

- The input starts with three integers, $0 \leq h \leq 10^6$, the amount of honey you have, and $1 \leq n, m \leq 10^3$, the dimensions of the grid.
- Then follow n lines, one for each row of the grid. Each row has m symbols, either `.`, representing an empty cell, or `#`, representing a filled cell. These symbols are separated by spaces. Furthermore, every second row starts with a space, as these are slightly offset to the right.

The grid always has enough open cells to store all your honey.

Output

Output a single integer, the number of cells you have to funnel honey into directly to store all your honey in the hive.

Example

Input	Output
8 4 4 . # # . # . # . # # # . # . . .	3
6 3 6 . . # # . . . . # # . # # # # . # #	2

Problem C. Criss-Cross Cables

Source file name:	crisscross.c, crisscross.cpp, crisscross.java, crisscross.py
Input:	`Standard`
Output:	`Standard`

As a participant in the BAPC (Bizarrely Awful Parties Competition) you are preparing for your next show. Now, you do not know anything about music, so you rip off someone else's playlist and decide not to worry about that any more. What you do worry about, though, is the aesthetics of your set-up: if it looks too simple, people will be unimpressed and they might figure out that you are actually a worthless DJ.

Picture by Oliver Quinlan on Flickr. ©BY-NC 2.0

It doesn't take you long to come up with a correct and fast solution to this problem. You add a long strip with a couple of useless ports, and add some useless cables between these ports. Each of these cables connects two ports, and these special ports can be used more than once. Everyone looking at the massive tangle of wires will surely be in awe of your awesome DJ skills.

However, you do not want to connect the same two ports twice directly. If someone notices this, then they will immediately see that you are a fraud!

You've made a large strip, with the ports in certain fixed places, and you've found a set of cables with certain lengths that you find aesthetically pleasing. When you start trying to connect the cables, you run into another problem. If the cables are too short, you cannot use them to connect the ports! So you ask yourself the question whether you're able to fit all of the cords onto the strip or not. If not, the aesthetics are ruined, and you'll have to start all over again.

Input

- The first line has $2 \leq n \leq 5 \cdot 10^5$ and $1 \leq m \leq 5 \cdot 10^5$, the number of ports on the strip and the number of wires.
- The second line has integers $0 \leq x_1 < \ldots < x_n \leq 10^9$, the positions of the n sockets.
- The third line has m integers $l_1, \ldots, l_m$, the lengths of the wires, with $1 \leq l_i \leq 10^9$.

Output

Print `yes` if it is possible to plug in all the wires, or `no` if this is not possible.

Example

Input	Output

`4 4`
`0 2 3 7`
`1 3 3 7`
	`yes`

`3 4`
`0 1 2`
`10 10 10 10`
	`no`

Problem D. Daily Division

Source file name:	daily.c, daily.cpp, daily.java, daily.py
Input:	`Standard`
Output:	`Standard`

Oostende Beach is a very long beach located in the north of Belgium. On this beach, there are n huts located along a straight line. People can rent a room in one of those huts to spend their beach vacations together with the other tenants.

Picture by biskyt30 on Flickr. ©BY-NC-ND 2.0

Every day at lunch time, a food truck rides by to serve fries to the guests. The truck will park in front of one of the huts and people will form two queues. The people staying in huts to the left of the food truck will queue on the left, and the people to the right of the food truck will queue together on the right. The people staying in the hut in front of the food truck will split their group in half, one half going to the left queue and the other half going to the right queue. If this is an odd number of people, the remaining person will go to the queue with fewer people, or choose one randomly if the queues have the same length. The food truck will always position itself so that the difference between the number of people in the left queue and the number of people in the right queue is as small as possible.

Each night the number of guests in exactly one of the huts changes. Can you help the food truck find the best position for each day?

Input

- The first line of the input consists of two integers $1 \leq n \leq 10^5$, the number of huts, and $1 \leq q \leq 10^5$, the number of days.
- The second line has n integers $a_0, \ldots, a_{n-1}$ satisfying $1 \leq a_i \leq 10^6$ for $0 \leq i < n$, where a_i is the current number of people in hut i.
- Then follow q lines with two integers $0 \leq i < n$ and $1 \leq x \leq 10^6$. The j^{th} of these lines indicates that at day j the number of people in hut i changes to x.

Output

- Print q lines: the optimal position of the foodtruck after each of the q nights. If there are multiple optimal positions, print the smallest one.

Example

Input	Output

5 4
3 1 3 4 2
0 5
0 9
4 5
2 1
	2
	1
	2
	1

4 8
1 1 1 1
2 2
1 2
2 1
1 1
3 2
2 2
1 2
2 1
	1
	1
	1
	1
	1
	2
	2
	2

Problem E. Eating Everything Efficiently

Source file name:	eating.c, eating.cpp, eating.java, eating.py
Input:	`Standard`
Output:	`Standard`

Margriet A. is in pizza heaven! She has bought a oneday access pass to Pizza World. Pizza World is a food festival, where all stands have their own special type of pizza. Margriet would really like to try many different types of pizza, but she thinks that she can only eat two pizzas in total. Therefore, she has come up with a cunning plan: at each stall she visits she decides whether she wants to buy this pizza or not. At the first stall where she decides to make a purchase, she will buy and eat exactly one pizza. At the second one, she will buy and eat half a pizza, and at the third she will eat one quarter of a pizza, etc.. Therefore, at the kth stall where she decides to buy some pizza, she will eat $\frac{1}{2k-1}^{th}$ part of a pizza. This way she makes sure that she will never get full!

Picture by Ken Lund on Flickr.
©BY-SA 2.0

In order to ensure that the flow of people in the park is adequate, the pizza stalls are connected by one-way paths, and to make sure that everyone will leave the festival, it is made impossible to visit a pizza stall more than once. However, every stall can be reached from the stall at the entrance, which is the stall with number 0.

Of course, Margriet has her own taste: she will like some pizzas more than others. Eating pizza from a stall will give her a certain amount of satisfaction which is equal to Margriet's personal stall satisfaction number multiplied by the fraction of a whole pizza she eats there. Her total satisfaction is the sum of satisfactions of every stall she visits. Can you help Margriet plot a route between the pizza stalls that satisfies her the most?

Input

- Two integers $1 \leq n \leq 5 \cdot 10^5$ and $0 \leq m \leq 5 \cdot 10^5$, the number of pizza stalls and the number of one way connections.
- The second line has n integers $c_0, \ldots, c_{n-1}$, $0 \leq c_i \leq 10^9$, the amount of enjoyment Margriet gets from eating one pizza at stall i.
- The next m lines each contain 2 integers $0 \leq s < n$ and $0 \leq t < n$, indicating a one way path from stall s to stall t. No connection will appear twice in the input.

Output

- Print the maximal amount of enjoyment Margriet can reach at the pizza fair. Your answer will be considered correct if it differs from the actual answer by an absolute error of at most 10^{-6}.

Example

Input	Output

`5 5`
`1 4 6 2 100`
`0 1`
`1 2`
`0 3`
`2 4`
`3 4`
	`100`

`3 2`
`1 0 1`
`0 1`
`1 2`
	`1.5`

`3 2`
`3 2 1`
`0 1`
`1 2`
	`4.25`

Problem F. Floating Points

Source file name:	floating.c, floating.cpp, floating.java, floating.py
Input:	`Standard`
Output:	`Standard`

Your ship has sunk. You want it back.

The best way[1] to salvage a shipwreck is to release a whole bunch of ping pong balls beneath it. The ping pong balls float up to the surface, and their buoyancy takes the sunken ship up towards the surface with them. At least, they do that as long as the ping pong balls actually push the ship up: if they slide off the shipwreck, or float past it entirely, or even surface in an inside air bubble of the ship, they of course do not contribute to pushing the ship out of the water.

You have already released a bunch of ping pong balls beneath your ship, and you want to know how many of the balls actually contribute to the salvage. Fortunately, you know exactly where you have released the balls, and you have a very precise model of the ship. Can you compute how many of the balls are helping you out?

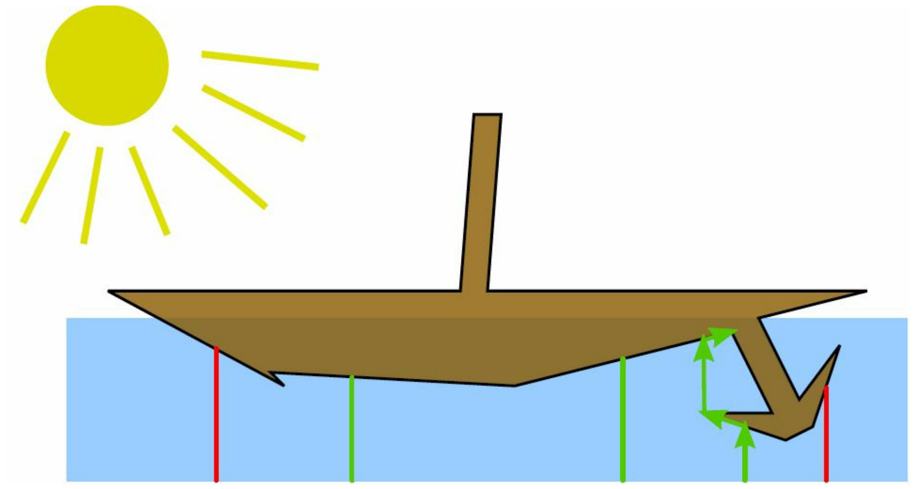


Figure 2: The shipwreck from sample 2. The fourth ping pong ball from the left, for instance, will help you out as it gets stuck between the anchor and the ship.

The sea is given as the (x, y)-plane. The surface of the sea is at $y = 0$, so that the water is where the y-coordinate is less than or equal to zero. The shipwreck is modeled by a 2D polygon in this plane. The ping pong balls are modeled as points (floating points) which are released far below the shipwreck.

A ping pong ball floats straight upwards, unless it surfaces or it is blocked by an edge. If it can float upwards along an edge, it will follow the edge. If it is blocked by a horizontal edge, or by a corner from which there is no edge going upwards, the ball is stuck.

A ball does not push the shipwreck up if it surfaces. If a ball gets trapped below the ship at $y = 0$, it still contributes to pushing the ship up, and hence is counted in the final answer.

Input

- The input starts with a line with integers $3 \leq n \leq 10^3$, the number of vertices of the shipwreck, and $1 \leq b \leq 2 \cdot 10^3$, the number of ping pong balls.
- Then follow n lines, each with two integers $-10^5 \leq x, y \leq 10^5$, which give the vertices of the polygon in counter-clockwise order.

[1]This does not work. Do not try this on your own sunken ship.

- Then one line follows containing b integers $x_1, x_2, \dots, x_b$, indicating the x-coordinates where the ping pong balls were released. It satisfies $|x_i| \leq 10^5$. The original y-coordinate of the ping pong balls is below -10^5.

The shipwreck is a simple polygon: its edges only intersect at vertices, and only two edges intersect at a vertex. Furthermore, no two x-coordinates, both of the points of the boat and of the balls, are the same.

Output

- The output consists of a single integer: the number of balls which end up pushing the shipwreck up.

Example

Input	Output

4 4
1 2
4 -4
6 0
9 0
2 3 5 8
	2

17 5
-28 2
-15 -5
-16 -4
2 -5
18 -1
21 -7
16 -7
22 -9
24 -8
26 -2
23 -6
20 0
28 2
0 2
1 15
-1 15
-2 2
-20 -10 10 19 25
	3

Problem G. Green Light

Source file name: green.c, green.cpp, green.java, green.py
Input: `Standard`
Output: `Standard`

Sarah is cycling to work. On her way there, she encounters the same traffic light every day. Before she reaches the lights, she alternates between using social media on her mobile device and glancing at the traffic lights, observing if they are green, yellow or red at that time. From experience, she knows that the lights have a fixed green-yellow-red cycle, and how long each stage lasts. So if the light goes from red to green at time T, she knows it will stay green until (but not including) $T + T_g$, then go yellow until (but not including) $T+T_g+T_y$ and finally stay red until (but not including) $T+T_g+T_y+T_r$, at which point it will turn green again. However, she does not know T, the time at which the traffic light cycle starts. Based on her observations, she can deduce what values of T are (im)possible. Assuming that each possible value of T that is consistent with her observations is equally likely, can you compute the probability that the lights will be green at a certain time?

Input

- The first line contains three positive integers T_q T_y T_r, corresponding to the duration (in seconds) for which the lights stay green, yellow, and red ($0 < T_g, T_y, T_r \leq 10^8$).
- The second line contains a single positive integer n, the number of times Sarah looked at the lights ($3 \leq n < 1000$).
- Each of the next n lines contains one integer $0 \leq t \leq 10^9$ followed by a color c : the time(in seconds) of the observation and color of the lights at that moment. Sarah did see the lights being each color (`green`, `yellow` and `red`) at least once.
- The last line contains an integer $0 \leq t_q \leq 10^9$ and a color c_q. These specify the question asked: What is the probability of the lights being of color c_q at time t_q?

Output

- $0 \leq p \leq 1$, the probability of the lights being color c_q at time t_q. Your answer will be considered correct if it differs from the actual answer by an absolute error of at most 10^{-3}.

Example

Input	Output
4 4 4	0.25
3	
2 green	
18 yellow	
34 red	
5 green	
4 4 4	0
4	
2 green	
6 yellow	
10 red	
14 green	
4 red	
6 6 6	1
6	
5 green	
6 green	
9 yellow	
12 yellow	
15 red	
19 red	
7 green	

Problem H. H to O

Source file name:	htoo.c, htoo.cpp, htoo.java, htoo.py
Input:	`Standard`
Output:	`Standard`

$$C_6H_{12}O_6 + 6O2 \rightarrow 6CO_2 + 6H_2O$$

Professor Cesium has created a new process to transform some chemical product into another type of chemical with some residues. The process is simple: he just needs to input a given number of molecules of type A, enter the output type B he desires and start the machine. It will create as many molecules of type B as possible. Unfortunately, professor Cadmium was jealous of his work and tried to sabotage the machine by inverting wires on his machine. Professor Cesium, alerted by one of his assistants, was able to repair the mistake. To detect any problem in the future, he is asking you to create an automatic way to compute the number of molecules that the machine should output. With this algorithm, he will be able to detect whether his precious machine was tampered with.

Molecules are written as string composed of upper case letters and numbers. Upper case letters represent atoms. Note that Cesium only uses single letters of the alphabet as abbreviations for atoms, so H, C, A, X, Y, ... can be used but He, Mg, ... can not. If a letter is not followed by a number, it means there is only one atom of it. An atom followed by a number l ($1 \leq l < 10^3$) represents l copies of that atom. Atoms can appear multiple times in a chemical product.

For example: H2OC100H means 2 atoms of H, then 1 of O, then 100 of C then 1 of H again.

Input

- The first line contains the input molecule, a string of length at most 2500, followed by an integer $1 \leq k \leq 10^3$, denoting how many of these molecules professor Cesium has.
- The second line contains the desired output molecule, given as a string of length at most 2500.

Output

- The output consists of a single line containing the maximum number n of output molecules we are able to construct using the input molecules.

Example

Input	Output

`H 2`
`O`
	`0`

`C2H6 10`
`C3H8`
	`6`

`CH3OH 1`
`CH4`
	`1`

`C6H6OCH2O 10`
`HCN`
	`0`

`C6H14 10`
`C5H10`
	`12`

`AB2CD1 2`
`A2B3CD2`
	`1`

Problem I. Isomorphic Inversion

Source file name:	isomorphic.c, isomorphic.cpp, isomorphic.java, isomorphic.py
Input:	`Standard`
Output:	`Standard`

Let s be a given string of up to 10^6 digits. Find the maximal k for which it is possible to partition s into k consecutive contiguous substrings, such that the k parts form a palindrome. More precisely, we say that strings $s_0, s_1, \ldots, s_{k-1}$ form a palindrome if $s_i = s_{k-1-i}$ for all $0 \leq i < k$.

In the first sample case, we can split the string 652526 into 4 parts as 6|52|52|6, and these parts together form a palindrome. It turns out that it is impossible to split this input into more than 4 parts while still making sure the parts form a palindrome.

Input

- A nonempty string of up to 10^6 digits.

Output

- Print the maximal value of k on a single line.

Example

Input	Output
652526	4
12121131221	7
123456789	1
132594414896459441321	9

Problem J. House Numbers

Source file name:	house.c, house.cpp, house.java, house.py
Input:	`Standard`
Output:	`Standard`

Peter was walking down the street, and noticed that the street had houses numbered sequentially from m to n. While standing at a particular house x, he also noticed that the sum of the house numbers behind him (numbered from m to $x-1$) equaled the sum of the house numbers in front of him (numbered from $x+1$ to n).

Given m, and assuming there are at least three houses total, find the lowest n such that this is possible.

Input

Input consists of a single line containing the integer m ($1 \leq m \leq 1,000,000$).

Output

On a single line, print m, x, and n, in order, separated by spaces.

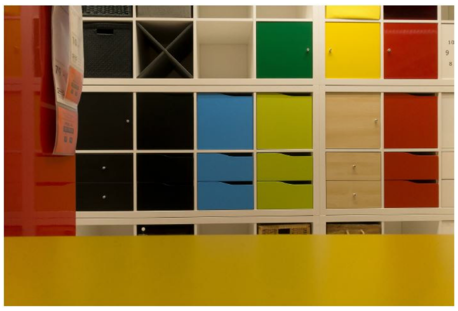
It is guaranteed that there will be a solution with n less than or equal to 10,000,000.

Example

Input	Output
1	1 6 8
11	11 49 68
999999	999999 1317141 1571535
999000	999000 1000000 1000999

Problem K. KALLAX Construction

Source file name:	kallax.c, kallax.cpp, kallax.java, kallax.py
Input:	`Standard`
Output:	`Standard`



You are the owner of IKEA, and you need to order a large number of bolts B. There is a single bolt manufacturer, but there are multiple companies reselling these bolts in packs (e.g. boxes, pallets). These companies form a directed chain, where each company buys packs from the previous company and combines these into new packs (with, of course, the logo of the company displayed brilliantly).

At first glance, it might seem that these intermediate companies offer no advantage, as they just repack the packs of the previous company into larger packs. However, every company has their own target audience, so they want to sell packs with a specific amount of bolts. Because every company only uses the packs of the previous company, it might not be possible to create a pack that has the exact number of bolts specified. Instead, if a company wants to create a pack which is guaranteed to contain X bolts, it will bundle various packs from the previous company where the displayed amount of bolts on these packs sums to no less than X. If there are multiple such combinations it will pick any from those whose displayed sum is minimal. For a better understanding, see the example below. Note that individual companies have no knowledge of the supply chain other than the pack sizes the previous company offers them.

You realise you can take advantage of this: when a company specifies that a pack has a certain number of bolts, it might in practice contain more! Therefore you start a quest of figuring out which pack has the lowest advertised amount, while still containing at least the number of bolts you need. Thanks to your business relations, you can freely choose the company to buy a pack from, including the manufacturer.

Explanation of first sample

Suppose that we would like to buy $B = 310$ bolts, and that there are three companies. The manufacturer (company one) sells packs of 40 and 65 bolts. Company two sells packs of 100 and 150 bolts. It cannot get these exact amounts from company one, and instead composes them as $100 \leq 40 + 65$ and $150 \leq 40 + 40 + 40 + 40$.

Next comes company three, offering packs of 300 and 320 bolts. It can assemble its 300-pack using three 100-packs (which we know will actually contain $105+105+105 = 315$ bolts) or using two 150-packs (which we know will actually contain $160 + 160 = 320$ bolts). However, for company three either combination is fine, so you do not know how many bolts a pack will actually contain. In this case you assume the worst, i.e. that this pack contains 315 bolts.

For its second pack of 320 bolts, company three will use $100 + 100 + 150 \geq 320$ (which we know really contains $105 + 105 + 160 = 370$ bolts). There are other combinations adding up to more than 320, but none achieve the minimum of 350, so we know company three will pick that combination.

Note in particular, that company three **does not know** that the 150-packs of company two actually contain 160 bolts (otherwise it could compose its 320-pack out of two of these). It only knows the amounts advertised by company two.

The packet of size 300 sold by company three is the smallest advertised packet that contains at least $B = 310$ bolts, so this is the packet we should buy.

	pack one		pack two	
	advertised amount	real amount	advertised amount	real amount
Company one	40	40	65	65
Company two	100	105	150	160
Company three	300	315 or 320	320	370

Input

- The first line of the input contains an integer $1 \leq B \leq 10^3$ giving the number of bolts that you need.
- The second line of the input contains an integer $1 \leq k \leq 10$ giving the number of companies.
- The next k lines each describe a company. Each line consists of the integers $l_i, n_1, n_2, \ldots, n_{l_i}$ meaning that the company i produces $1 \leq l_i \leq 10$ types of packages of sizes $0 < n_1 < n_2 < \ldots < n_{l_i} \leq 10^3$, respectively.

Output

- A single integer giving the smallest size of a package that you can buy which contains at least B bolts no matter how the companies build their packages, or `impossible` if this cannot be achieved.

Example

Input	Output

310
3
2 40 65
2 100 150
2 300 320
	300

371
3
2 40 65
2 100 150
2 300 320
	impossible

90
2
2 20 35
2 88 200
	88

91
2
2 20 35
2 88 200
	200

Problem L. Greedy Scheduler

Source file name:	scheduler.c, scheduler.cpp, scheduler.java, scheduler.py
Input:	`Standard`
Output:	`Standard`

A store has n cashiers numbered sequentially from 1 to n, with c customers in a queue. A customer at the front of the queue is constantly assigned to the first unoccupied cashier, *i.e.*, cashier with the smallest number. The i_{th} customer's shopping cart takes t_i seconds to process.

Find which cashier will process each customer's shopping cart.

Input

The first line of input contains two space-separated integers n and c ($1 \leq n \leq c \leq 1000$). The second line of input contains c space-separated integers $t_1, ..., t_c$, representing the length of time required to handle that customer.

Output

Output a single line containing c space-separated integers, each with the cashier number that handles that customer.

Example

Input	Output
3 10 406 424 87 888 871 915 516 81 275 578	1 2 3 3 1 2 3 1 2 1