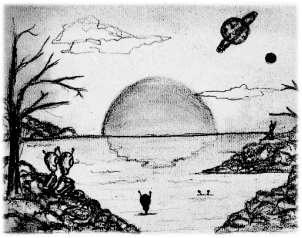
Problem A. Alien Sunset

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



Following tremendous advances in space flight control software and equally impressive innovations in reality TV crowdfunding, humans have successfully settled a number of planets, moons, asteroids, and various other kinds of funny-shaped rocks across our solar system.

To celebrate, the Galactic President has elected to create a new holiday called "Solar Night". At the crux of the event, she decrees, every settlement will simultaneously launch colourful fireworks into the darkness of night.

Night, like most things, is a difficult problem in space. Every spacebound object has its own day length and period of rotation. Thankfully all of the settlements did, at least, start their clocks at the same moment. Settlements may have started in daylight or darkness and so it is possible that the first recorded sunrise can be either before or after the first hour of sunset.

By convention, the President's term lasts for exactly 1825 days as measured by the planet with the longest period of rotation. The celebration needs to happen within that time or it will not serve its intended purpose.

Determine how many hours must pass for us to find a suitable time to celebrate Solar Night.

Input

- One line containing the integer N ($1 \leq N \leq 20$), the number of settlements.
- N lines, each containing three integers:
  - H ($2 \leq H \leq 100$), the number of hours in this settlement's solar day.
  - R and T ($0 \leq R, T \leq H - 1, R \neq T$), the hours of sunrise and sunset respectively.

At sunrise and sunset, a settlement is in darkness. At times strictly in between sunrise and sunset, a settlment is in daylight.

Output

Output the number of hours that must pass from when the settlement clocks began until each settlement is in darkness. If no suitable time occurs in the first 1825 days, output `impossible`.

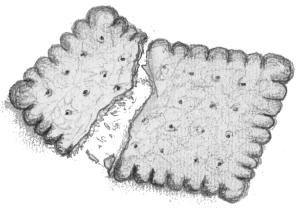
Example

Input	Output
2 24 7 19 24 18 6	6
3 10 8 2 15 5 10 20 15 10	12
2 6 4 2 12 7 5	impossible
2 10 5 6 10 6 5	5

Problem B. Breaking Biscuits

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

This year, Walter's workplace resolved to try something radically different: they're going to change the weekly order of biscuits for the break room to a whole other brand.



Biscuits come in many shapes and sizes, but the particular brand they settled on has two special qualities:

- It is completely planar (two-dimensional);
- It is perfectly polygon-shaped.

However, disaster struck immediately: the available mugs in the break room are too narrow for Walter to be able to dunk these new biscuits into, no matter what combination of rotations along the three axes he tries.

There is only one thing for it: Walter will need to order another mug.

Before taking this drastic step, it is vital to know how wide the diameter of this mug will need to be in order to succesfully accommodate a (possibly rotated) biscuit of the new brand.

Input

- One line containing an integer N ($3 \leq N \leq 100$), the number of vertices in the biscuit.
- Each of the following N lines contains two space-separated integers X_i and Y_i ($-10^5 \leq X_i, Y_i \leq 10^5$), the coordinates of the i-th vertex.

Vertices are always given in anti-clockwise order. Also, as anyone can tell you, biscuits never self-intersect and always have positive area.

Output

Output the minimum possible diameter of the new mug, in order that it can fit the new kind of biscuit fully inside in at least one orientation. The output must be accurate to an absolute or relative error of at most 10^{-6}.

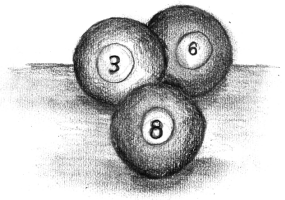
Example

Input	Output
4 0 0 5 0 5 2 0 2	2.0
6 81444 14017 80944 13517 81127 12834 81810 12651 82310 13151 82127 13834	1224.7089450046291
8 197 239 208 246 221 241 250 254 220 265 211 258 198 268 163 256	28.816782

Problem C. Cued In

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

Snooker is a cue sport played by two players on a rectangular table. The players take turns to pot a series of balls of varying colours, where each colour represents a distinct point value for potting the ball.



A player may pot any ball on the table initially, however any subsequent shots must follow a pattern: if the previous ball was red, the next ball must be another colour; otherwise, if there are still red balls left, the next ball must be red.

Balls of any colour other than red are initially replaced on the table every time they are potted, and may be used again to score more points. The balls stop being replaced once all of the red balls have been potted.

The values of each coloured ball are:

Colour	red	yellow	green	brown	blue	pink	black
Value	1	2	3	4	5	6	7

Snooker players are respected universally for their prowess in mental arithmetic. One sweeping glance across the table is enough to tell an experienced contestant how much they could score.

For newer players, however, this is a challenge. Write a program to help calculate a score for a given list of balls remaining on the table.

Input

- one line containing the integers N ($1 \leq N \leq 21$), the number of balls remaining on the table.
- N further lines, each containing the colour of one of the balls on the table.

The list of balls will not be ordered in any way and will contain at most one of each of yellow, green, brown, blue, pink and black.

Output

Output the largest possible score the player can make.

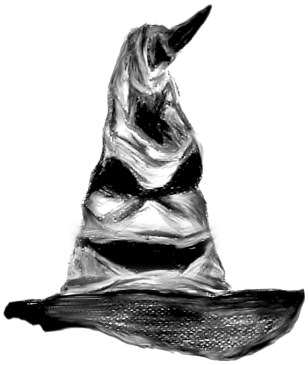
Example

Input	Output
5 red black pink red red	37
3 blue black pink	18
8 yellow green brown red red red red red	34
2 red red	1

Problem D. Deranging Hat

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

In the wizarding world of security, there are two kinds of researcher: the idealist *arranging hat* and the mercenary *deranging hat*.



As we learned last year, an *arranging hat* carefully sorts out any list of letters given to it into ascending order. However, a *deranging hat* performs the exact opposite function: putting a sorted string of letters back into its original order.

The tool of choice for today's discerning headwear is a sorting network: a sequence of instructions represented by a list of pairs of numbers A_i and B_i, meaning that if at step i the A-th item in the string is not already smaller than the B-th item, they should be swapped immediately.

Given a specific word W, output a sorting network that the deranging hat can use to form the word from its original sorted letters.

Input

One line containing one string of lowercase Latin letters ('a'-'z'), S, containing at most 1000 characters.

Output

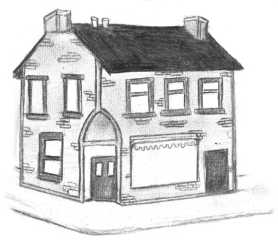
Output at most 10000 lines, each containing two integers A_i and B_i ($1 \leq A_i, B_i \leq |S|$) giving the i-th operation to perform.

Example

Input	Output
bab	2 1
dude	4 3 3 2

Problem E. Education

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



Seeking to cash in on the lucrative private education business, *EduCorp* recently established the prestigious "Bootcamp Academy of Economics" and, counter to their early projections, is growing rapidly.

So rapidly, in fact, that the student body is already overflowing the small (but prestigious) campus building and now needs to be contained somewhere else while more new (and prestigious) buildings are built.

Each department will sell off its original space and then move into its own new rented building. As departments are deeply territorial, buildings must not be shared. Because this is an economics academy, the capacities and rents of each of all the local available buildings were easy to find by disguising the task as homework.

However, it still remains to choose which buildings to rent so as to minimise total budget. This is where you can help.

Input

- one line containing the integers n and m ($1 \leq n \leq m \leq 5000$), the number of departments and buildings respectively.
- one line containing n integers $s_1 \ldots s_n$ ($1 \leq s_i \leq 1000$ for each i), where s_i is the number of students in department i.
- one line containing m integers $p_1 \ldots p_m$ ($1 \leq p_i \leq 1000$ for each i), where p_i is the capacity of building i.
- one line containing m integers $r_1 \ldots r_m$ ($1 \leq r_i \leq 1000$ for each i), where r_i is the yearly rental cost of building i.

Output

If it is not possible to rent enough buildings for all the departments, output `impossible`.

Otherwise, output n unique, space-separated integers $v_1 \ldots v_n$, where the i-th number is the building to be rented by the i-th department so as to minimise the total spend on rent. If there are multiple equally good answers, you may print any.

Example

Input	Output
2 5 40 200 1000 199 201 10 50 600 300 400 200 800	2 3
3 5 10 20 30 30 25 20 15 10 30 25 20 15 10	5 3 1
1 1 20 10 1	impossible

Problem F. Flipping Coins

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

Here's a jolly and simple game: line up a row of N identical coins, all with the heads facing down onto the table and the tails upwards, and for exactly K times take one of the coins, toss it into the air, and replace it as it lands either heads-up or heads-down. You may keep all of the coins that are face-up by the end.

Being, as we established last year, a ruthless capitalist, you have resolved to play optimally to win as many coins as you can. Across all possible combinations of strategies and results, what is the maximum expected (mean average) amount you can win by playing optimally?

Input

- One line containing two space-separated integers:
  - N ($1 \leq N \leq 400$), the number of coins at your mercy;
  - K ($1 \leq K \leq 400$), the number of flips you must perform.

Output

Output the expected number of heads you could have at the end, as a real number. The output must be accurate to an absolute or relative error of at most 10^{-6}.

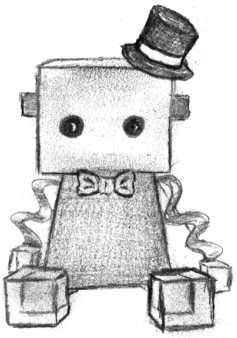
Example

Input	Output
2 1	0.5
2 2	1
2 3	1.25
6 10	4.63476563
6 300	5.5

Problem G. GentleBots

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

Rainforest Inc. is opening a large new automated warehouse in the far Northern reaches of the UK—some place they call "Walthamstow".



The robotic worker drones inside will operate in not just one or two, but three dimensions, and can move in any one of the 6 cardinal dimensions in steps of 1 metre at a time. For example, a robot looking to move from position (X_1, Y_1, Z_1) to position (X_2, Y_2, Z_2) will, assuming no obstacles, need to take $(|X_2 - X_1| + |Y_2 - Y_1| + |Z_2 - Z_1|)$ steps in total.

Since this warehouse is in Britain, and because every stereotype is true, the robotic denizens are all impeccably polite. When two robots travelling in opposite directions meet, they wordlessly negotiate for one of the robots to step aside somehow so the other can pass.

Multiple robots cannot occupy the same integer co-ordinates, and no two robots can swap positions in one move. All moves are instantaneous.

We have prepared a test run of the warehouse with just two machines installed. Write a program to pass this test.

Input

Two lines, one for each robot, each containing six space-separated integers $(X_0 Y_0 Z_0)$ and $(X_\infty Y_\infty Z_\infty)$, the intended start and end locations of a robot respectively $(-1000 \leq X, Y, Z \leq 1000)$.

The robots will start in different places from each other, and will also end in different places from each other.

Output

Output up to 7000 lines, giving one possible list of locations of the robots over time. The position of both robots at time T must be given as bracketed space-separated (X, Y, Z) co-ordinate tuples on line T.

Co-ordinates must not exceed an absolute value of 10^6.

Example

Input	Output
0 0 0 2 2 2 1 1 2 0 0 0	(0 0 0) (1 1 2) (1 0 0) (1 1 1) (1 1 0) (0 1 1) (1 1 1) (0 1 0) (1 1 2) (0 0 0) (1 2 2) (0 0 0) (2 2 2) (0 0 0)
-2 0 0 1 0 0 3 0 0 -1 0 0	(-2 0 0) (3 0 0) (-1 0 0) (2 0 0) (0 0 0) (1 0 0) (1 0 0) (1 0 -1) (1 0 0) (0 0 -1) (1 0 0) (-1 0 -1) (1 0 0) (-1 0 0)
0 0 0 1 0 0 1 0 0 0 0 0	(0 0 0) (1 0 0) (0 1 0) (0 0 0) (1 1 0) (0 0 0) (1 0 0) (0 0 0)

Problem H. Hiker Safety

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

Times are hard at the Association of Chartered Mountaineers.

The growth of their pedestrian sport has slowed to a crawl. Instead of taking up mountaineering, younger potentials are gravitating towards warmer indoor activities such as snooker, musical chairs, and programming contests.

To win over more members, the Association is going to organise a series of new time trial orienteering events next year. The route for the first race will be a short run through the Cairngorms, with every contestant following the same route designated by marker points but all starting at different times.

Because hiking can be dangerous, and many of the contestants will be inexperienced, the competition committee drew up two rules:

- Every contestant needs to keep a specific maximum distance away from the next-closest contestant, in either direction, at all times.
- Every contestant should be given personal space. If a contestant needs a personal space of D metres, nobody else should ever come closer than that at any time. This distance varies according to level of experience.

The hardest part of orienteering is the pathfinding; once a contestant knows where to go next, they can get there in almost no time (for the purposes of this problem, instantaneously).

In fact, while the inaugural ACM "Icy-Cold Peak Contest" is already underway, pathfinding is turning out to be a problem: nobody is sure whether they can move next without breaking any of the conditions on minimum and maximum distance.

Help the runners reach the end of their route by computing a list of who should move to the next goal point, and at what time.

Input

- One line containing an integer B ($1 \leq B \leq 50000$), the maximum separation allowed between any two runners.
- One line containing an integer P ($3 \leq P \leq 1000$), the number of marker points along the route.
- One line containing P unique space-separated integers $d_1 \ldots d_P$i ($0 \leq d_P \leq 10^6$), with d_i being the distance of the i-th vertex from the starting point $d_1 = 0$.
- One line containing an integer K ($2 \leq K \leq 1000$), the number of hikers on the landscape.
- K further lines, each containing the pair of space-separated integers A_i and V_i ($1 \leq A_i \leq 10^6; 1 \leq V_i \leq P$), the minimum separation distance and current marker position respectively for the i-th person.

The initial configuration of hikers will be legal according to the minimum and maximum distance rules. The hikers will be given in increasing order of distance from the start.

Output

If it is not possible to get everyone to the end of the route without breaking minimum or maximum distance requirements, output `impossible`.

Otherwise, output a space-separated list of moves on one line, each describing which person should make the next move.

If anyone falls off the landscape, your answer will not be judged as correct. However, once someone has arrived at the end of their journey they cease to count towards any rule violations.

Example

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

Input
10 10 0 1 3 6 10 14 17 19 20 21 3 3 1 1 3 3 5
Output
2 1 1 3 2 1 3 2 1 3 3 2 1 3 2 2 1 2 1 1 1

Input
5 5 0 2 5 9 14 2 2 1 2 2
Output
impossible

Problem I. I Work All Day

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

Michael is a lumberjack, and a pretty OK one at that. However, automation is making fast (if polite) inroads to his business, and he needs to stay ahead to remain competitive.

To this end, he has invented a machine called the *Flannelmaster GTX*. This is a fearsome tree-cutting contraption, powered by logarithms, which swings an axe horizontally from a specified height above the ground. Each impact of its electronic axe cuts the tree cleanly into two parts, the stump rolling away and the remainder of the tree falling down into the same place.

This continues until the remaining height is too small to cut any more, at which point any irregular-sized stump is discarded as waste and the remaining lumber of the same length as the setting is neatly packaged and sold on automatically.

What setting should Michael use for the height in order to minimise loss?

Input

- One line containing the integer N ($2 \leq N \leq 10$), the number of settings;
- One line containing N distinct integers H_i ($1 \leq H \leq 500$), the possible settings;
- One line containing the integer T ($1 \leq T \leq 3000$), the height of the tree.

Output

Output the best setting to use so that tree waste will be minimal. If there are multiple settings that yield equally small waste, you may print any one of them.

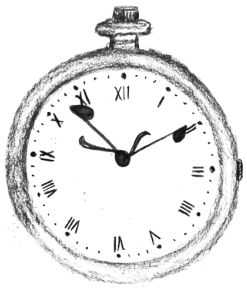
Example

Input	Output
3 5 6 8 103	6
4 7 3 5 13 1366	7

Problem J. Just A Minim

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

Listening to music is a good way to take one's mind off the monotony of typing out solution after correct solution.



However, it can be very annoying to start listening to a tune and then for time to run out early, cutting the listening short. How much more satisfying it would be if you could choose tunes to fit the time available!

With this in mind, you have found a way to code musical scores into simple lists of numbers representing the length of the notes in each tune as follows:

Code	Name	Length
0	breve	2 notes
1	semibreve	1 notes
2	minim	$\frac{1}{2}$ note
4	crotchet	$\frac{1}{4}$ note
8	quaver	$\frac{1}{8}$ note
16	semiquaver	$\frac{1}{16}$ note

Given such a list of numbers, calculate the length of the tune in notes.

Input

- One line containing the integer N ($1 \leq N \leq 2000$), the number of values in the tune.
- one line containing N integers each representing the length of a value using the codes above.

Output

Output the length of the tune, as a real number of notes. The output must be accurate to an absolute or relative error of at most 10^{-6}.

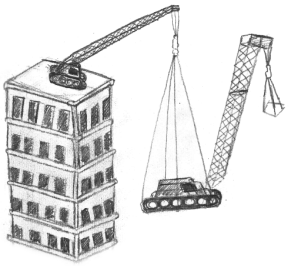
Example

Input	Output
4 1 1 1 0	5
10 1 2 2 2 1 4 4 8 8 16	4.3125

Problem K. Knightsbridge Rises

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

High-rise buildings in the wealthy retail district of Knightsbridge are usually built with exotic hoisting machines known, in construction circles, as *cranes*.



While it is common to mount these devices on the ground, it's not always ideal—for example, a skyscraper building would need an equally tall crane. In these cases, the clever solution the industry has developed is for a smaller crane to be mounted directly on top of the tower.

However, this solution presents another challenge: how can such heavy equipment be brought to the summit in the first place? The industry's solution is to simply use a smaller crane to lift the main crane up. And if that smaller crane is still too massive, find another smaller still, and so on, until said device fits inside a pocket and can be carried up by some enterprising engineer.

Once on top of the building, whether by being carried up, or by being lifted up by another crane capable of holding its weight, a crane can be used to lift others up onto its tower. Once raised, it is not possible to transfer a crane anywhere else.

We have several construction projects in progress at the moment, each with its own requirements on how heavy the loads eventually lifted from the ground need to be. Find an assignment of cranes to buildings that can satisfy these requirements.

Input

- One line containing an integer N ($1 \leq N \leq 100$), the number of types of cranes under our command.
- N further lines, each containing a pair of space-separated integers W_i and L_i ($0 \leq W_i, L_i \leq 10^6$), the weight and maximum lifting weight of the i-th crane in kilograms.
- One line containing an integer M ($1 \leq M \leq 100$), the number of tower blocks.
- One line containing the M space-separated integers T_i ($1 \leq T_i \leq 10^6$), T_i being the weight we need to be able to lift from the top of the i-th building by the end.

Output

If it is not possible to supply all of the buildings, output `impossible`.

Otherwise, output M lines. The i-th line should contain a list of space-separated integers $x_1 \ldots x_k$ ($1 \leq x \leq N$), describing the order in which cranes should be raised onto the i-th building from the input.

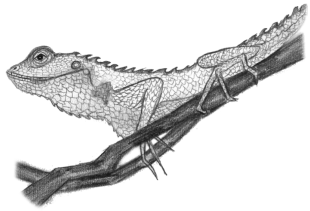
Example

Input	Output
5 0 1 1 2 2 3 3 4 0 2 2 4 2	5 3 4 1 2
7 0 1 1 4 0 3 0 1 2 5 2 5 1 2 3 5 4 5	3 5 1 2 4 7 6
2 0 1 5 3 2 2 1	impossible

Problem L. Lounge Lizards

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

Monitor lizards are a kind of reptile known mainly for their cold-bloodedness and addiction to computer screens. Due to their love for digital displays, these scaly creatures spend most of their time at home glued to a small television in the lounge.



Conflict has arisen at one particular reptile house. The audience here has grown so large that not everyone will be able to see the screen at once any more; specifically, a lizard will only be able to see enough if it is strictly taller than all of the lizards sitting exactly along the straight line from itself to the television.

Monitor lizards aren't particularly picky about the actual contents of the screen or being able to see it obliquely (or even from the front)—they just want to be able to keep an eye on it.

The lizards don't want to move, however. It's possible to chase a monitor lizard away in order for the ones behind it to see, or leave it alone, but re-homing somewhere else in the room is unthinkable.

Assuming lizards are removed optimally, how many at most can remain and still see the screen?

Input

- one line containing the space-separated integers T_X and T_Y ($-10^6 \leq T_X, T_Y \leq 10^6$), the co-ordinates of the television.
- one line containing the integer N ($1 \leq N \leq 10^6$), the number of lizards.
- N further lines, each containing three space-separated integers $X_i Y_i H_i$ ($-10^6 \leq X, Y \leq 10^6; 1 \leq H \leq 10^6$), the co-ordinates and height respectively of one lizard.

The co-ordinates of all televisions and lizards will be distinct.

Output

Output the maximum number of lizards that can stay and watch television at once.

Example

Input	Output
50 50 2 60 50 1 65 50 2	2
50 50 3 60 55 1 70 60 1 40 45 1	2
-100 0 6 -99 1 2 -98 2 4 -97 3 3 -96 4 4 0 100 3 100 0 7	4