Problem A. Awkward Auction

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

Your local warehouse's marketing department has thought of a new way of extracting money from their consumers: organizing the Battle in Assessing Precisely Costs (BAPC).

The coveted BAPC Cup.
© GEWIS on 2022.bapc.eu, used with permission

A battle is played against an auctioneer. The auctioneer has an unlimited number of identical gems available, which all have the same fixed secret worth. This secret worth is an integer between 1 and n (inclusive, in euros). In each round, you have to bid an integer amount of euros on such a gem, and your goal is to bid exactly the secret worth. If you bid at least the secret worth, you have to buy the gem for the amount of your bid. On the other hand, if you bid less than the secret worth, you do not get the gem and keep your bid. However, if you bid less than the worth, the auctioneer will give an endless speech why the gem is clearly worth more than the bid. To stop this speech and be able to make a new bid, you have to bribe the auctioneer with b euros each time. Finally, if you bid exactly the secret worth, then in addition to buying the gem, you get a nice cup showing that you won the battle, and the battle immediately ends.

Since the BAPC is your favourite competition, you of course want this cup! Therefore, you keep bidding until you bid the right amount and get the cup. You wonder how much this will cost you in the worst case, assuming that you make optimal decisions for the amounts to bid.

Input

The input consists of:

- One line with two integers n and b ($1 \leq n \leq 400$, $1 \leq b \leq 10\,000$), the given maximum worth of the gem and the bribe you need to pay when bidding too low.

Output

Output the maximum cost in euros that you have to pay in the worst case if you bid optimally.

Example

Input	Output
4 2	6
8 3	16

Problem B. Boat Commuter

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

The Bulgarian city of Nodnol runs a boat service to ferry its residents between the trendy areas in which they live and the large metallic structures in which they work on the next recession.

TFN (Transport For Nodnol) has issued m travel cards (known affectionally as “Retsyo”), which are numbered from 1 to m. Each pier has a card terminal at which passengers are required to tap “in” when starting the trip and to tap “out” when finishing it.

As there is only one card terminal on each pier, passengers use the same device to tap in and to tap out.

Trip cost depends on the distance travelled and is determined as follows:

- if the trip started at the pier i and finished at the pier j ($i \neq j$), then its cost is $|i - j|$ pounds;
- if the trip started somewhere and was not finished with a tap out, then it costs £100;
- if the trip started and finished in the same place, then it also costs £100, as it is interpreted as an attempt to game the system.

You are given a sequence of tapping events — for each you have the pier p_i and card number c_i recorded. You are to determine how much the transport authority should charge each of the cards

Input

- One line containing three integer numbers: the number of piers n, the number of travel cards m, and the number of events k ($2 \leq n \leq 50$, $1 \leq m, k \leq 10^5$).
- k further lines, each describing tap events in chronological order.
  - The i-th event is described by two integers p_i and c_i ($1 \leq p_i \leq n$, $1 \leq c_i \leq m$).

Output

Output m integers separated by spaces — the i-th integer giving the total charge to be applied to the i-th card.

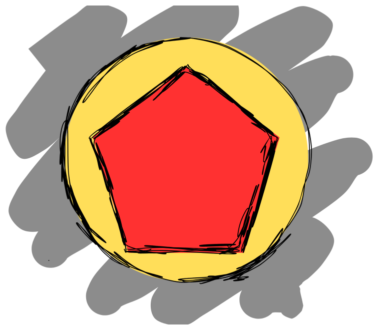
Example

Input	Output

3 3 5
1 1
1 2
1 2
3 1
2 3
	2 100 100

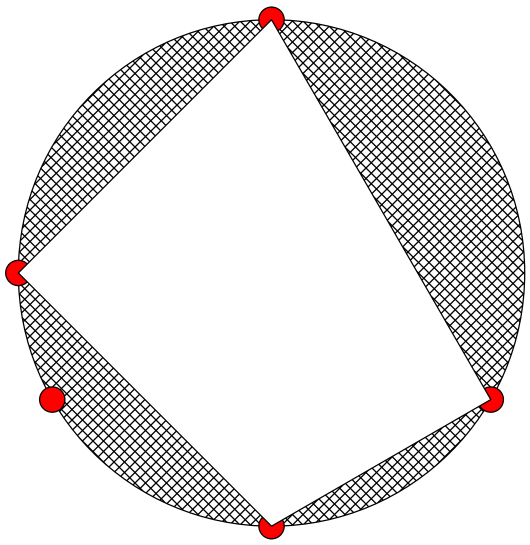
Problem C. Clearing Space

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



You are putting up an event space in Nottingham's Sherwood Forest by erecting a fence in a circular-shaped clearing you found that is exactly one kilometre in radius. You will put some fence posts in the trees around the edge of the clearing and then connect them together with fencing later.

You would like to put the fence around as much of the event space as possible. However, the ground is only suitable in a few places around the border, and you only have so many fence posts to put in the ground, so you'll have to choose carefully if you want to maximise area.



An illustration of using 4 posts to capture the maximum area in example input.

Knowing the safe places to put fence posts, and the number of posts you have, what is the maximum area of clearing you can enclose?

Input

- One line containing the integer number of safe points around the 1km-radius clearing, n ($3 \leq n \leq 100$).
- One line containing the integer number of fence posts you have, p ($3 \leq p \leq n$).
- One line containing n distinct real numbers $a_1, \ldots, a_n$ in ascending order, the angles in degrees of each of the safe places to add fence posts ($0 \leq a_i < 360$).

Output

Output the maximum area you can capture with a polygonal clearing made using at most p fence posts, in square metres.

The output must be accurate to an absolute or relative error of 10^{-6}.

As a reminder, the radius of the clearing is 1km.

Example

Input	Output

`5`
`4`
`0 120 180 240 270`
	`1866025.40378443866`

Problem D. Delivery Forces

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

Gry finally becomes the Executive Courier Officer in “Universe Express”. He has n subordinate couriers with some delivery strength f_i. The delivery strength of a team of three people is the median of their strength, i.e., the middle element after the sorting. Please help Gry to split the couriers into k teams of three people in order to maximize the total delivery strength of “Universe Express”. The total strength is the sum of the strength of these k teams.

Input

- One line containing the number of couriers in the company, n ($1 \le n \le 10^6$), where n is a multiple of 3.
- One line containing the strengths of the n couriers $f_1 \dots f_n$ ($1 \le f \le 10^6$).

Output

The sole line of the output should contain the maximal strength of “Universe Express”.

Example

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

Problem E. Eurokod

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

This year, for the first time, the *Eurokod* is being held, an international competition in writing beautiful and readable code!

There are n contestants participating in the competition, labeled with numbers from 1 to n, and each of them has written a code.

Their codes are evaluated by an association of computer scientists. The association consists of a president and members of the association. The president awards points to codes in one way, and the members of the association award points in another way.

President's points:

The president will rank the codes from the most beautiful to the least beautiful (in his opinion). The first code will be awarded n points, and each subsequent code will be awarded one point less than the previous one.

Members of the association's points:

Each member of the association will vote for the code he considers the most beautiful. After each member of the association has voted, the codes will be ranked in descending order according to the number of votes they received from the members of the association. The first code (the one with the most votes) will be awarded n points, and each subsequent code will be awarded one point less than the previous one.

Total points:

The total number of points for each code is equal to the sum of the points awarded by the president and the number of points awarded by the members of the association.

Your task is to print the order of codes in descending order according to the number of points.

If more codes have the same number of points, then the better ranked one is the one that has won more points from the members of the association.

Input

The first line contains an integer n ($1 \leq n \leq 50$), the number of contestants.

The second line contains n integers a_i ($1 \leq a_i \leq n$), where the i-th integer represents the label of the code that the president ranked i-th. The ranking of the president is given in the order from the most beautiful to the least beautiful, it contains all the labels from 1 to n exactly once.

The third line contains n integers b_i ($0 \leq b_i \leq 200$), where the i-th integer represents the number of votes that the i-th code received from the members of the association. There won't be two codes that received the same number of votes.

Output

In n lines, print the ranking of codes in descending order according to the number of points.

Each line should be in the form "`[rank]. Kod[label] ([number of points])`", where `[rank]` is the rank of the code in the ranking, `[label]` is the label of the code written in two-digit form with leading zeros, and `[number of points]` is the number of points that the code won.

For example, if the first place was won by the code with the label 3 with 12 points, then the first line is

“`1. Kod03 (12)`”.

Example

Input	Output

`3`
`1 2 3`
`50 10 20`
	`1. Kod01 (6)`
	`2. Kod03 (3)`
	`3. Kod02 (3)`

`5`
`5 2 4 1 3`
`4 5 2 1 3`
	`1. Kod02 (9)`
	`2. Kod05 (8)`
	`3. Kod01 (6)`
	`4. Kod04 (4)`
	`5. Kod03 (3)`

`7`
`6 3 2 1 5 4 7`
`200 56 11 0 13 105 12`
	`1. Kod06 (13)`
	`2. Kod01 (11)`
	`3. Kod02 (10)`
	`4. Kod03 (8)`
	`5. Kod05 (7)`
	`6. Kod07 (4)`
	`7. Kod04 (3)`

Explanation

Clarification of the first example:

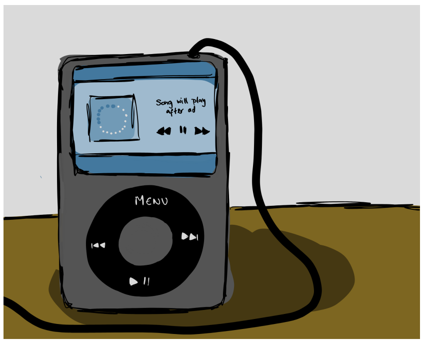
`Kod03` and `Kod02` have the same number of points, but `Kod03` has more votes from the members of the association, so it is better ranked.

Clarification of the second example:

The president ranked the `Kod05` as the most beautiful, so it won $n = 5$ points.

Problem F. Fast Forward

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



Gry has started to use the new Expify song streaming platform. Since, Gry does not want to spend money Expify forces him to listen to advertisements. An advertisement can be played only after some song (it cannot be played in the middle) and only if the time from the end of the previous advertisement is at least c seconds.

Gry has a circular playlist with n songs where the duration of the i-th song is d_i seconds. He wants to minimize the number of advertisements, so, he wants to find out how many advertisements will be if he starts listening to his whole playlist from i-th song, i.e., the circular playlist stops playing after n songs.

We suppose that there is an advertisement right before Gry starts listening. Neither this advertisement nor the one, after the playlist stops, count.

Input

- One line containing the number of songs in the playlist n, and the refresh time between advertisements c ($1 \leq n \leq 10^6$, $1 \leq c \leq 10^9$)
- One line containing the n durations of the songs $d_1 \dots d_n$ ($1 \leq d_i \leq 10^3$)

Output

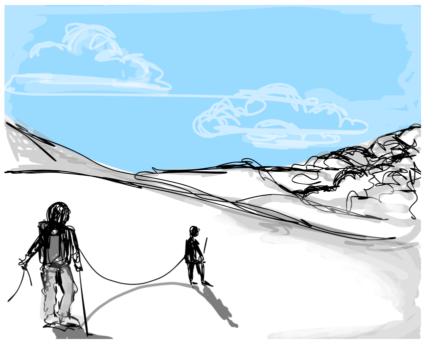
Output the number of advertisements if Gry starts listening to the playlist from the i-th song.

Example

Input	Output
7 7 1 1 1 1 1 1 1	0 0 0 0 0 0 0
3 3 1 1 3	0 1 1

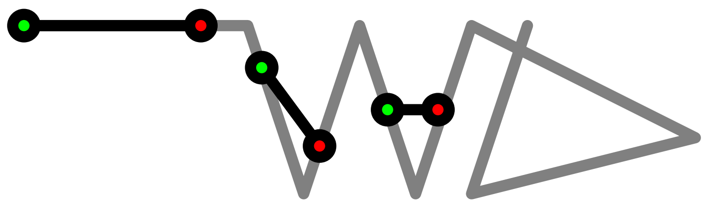
Problem G. Glacier Travel

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



Glaciers are vast rivers of slowly-flowing ice, fraught with crevasses which hide under thin layers of snow and wait for unsuspecting walkers to step into and fall in. To reduce the danger, hikers usually go in teams tied together with a thick rope to reduce the consequences of a fall–if one person falls in, the other person may yet hold them from a safe distance.

Today, you are roped up to cross a glacier with your partner. Your plan is to follow the exact same route, at the same speed, the first starting earlier and the second beginning to trace steps once you are exactly x metres apart. Were you to follow a completely straight path, you would thus then remain exactly x metres apart at all time.



An illustration of the path taken in the 2nd example case, taken from above. This could also be a particularly festive diagram of someone falling into a crevasse.

However, the twisting nature of the course as you avoid obstacles means that you may not always remain exactly x metres apart. What is the closest that you shall actually come while both of you are walking on the path?

Input

- One line containing a real number: the separation distance along the path in metres, s ($1 \leq s \leq 1000$).
- One line containing the number of points in the path, n ($2 \leq n \leq 10^6$).
- n further lines, the ith of which contains a pair of integers giving the ith coordinate on the track $x_i y_i$ ($-10^6 \leq x, y \leq 10^6$) in metres from the origin.

Every pair of adjacent points on the track are distinct from one another, although the track may cross over or repeat itself. The track is guaranteed to have a length of at least s.

Output

Output the minimum distance between the two walkers at any point on the route, ignoring any time after the first walker has finished, or before the second walker has started.

The output must be accurate to an absolute or relative error of at most 10^{-4}.

Example

Input	Output

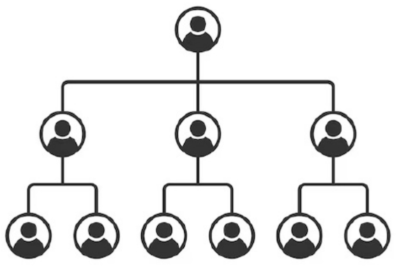
5
4
20 0
10 0
10 10
0 10
	3.5355339

3.16227766
9
-2 4
2 4
3 1
4 4
5 1
6 4
10 2
6 1
7 4
	1

Problem H. Hierarchy

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

Kresimir has started studying corporate structures, including *hierarchies*. He observed employees and their relationships within a company. In this case, we are only looking at *superior-subordinate* relationships, meaning relationships where one employee is directly superior to another employee in the company.



A hierarchy is a structure with N employees and $N-1$ superior-subordinate relationships, where there is one person who is directly or indirectly superior to all employees. In the observed company, there are also N employees and $N-1$ such relationships, but it is not certain whether this is a valid hierarchy or not.

Kresimir has asked you to help answer this question. He has recorded all the data in his notebook. Additionally, in his notebook, he will make Q permanent changes by reversing one superior-subordinate relationship such that the subordinate becomes superior to their former superior. After each such change, it is necessary to answer the same question: is the current state a valid hierarchy?

Input

In the first line, there is a positive integer N $\left(2 \leq N \leq 3 \cdot 10^5\right)$.

In the next $N-1$ lines, for each $i = 1, 2, \ldots, N-1$, there is a pair of integers p_i and e_i $(1 \leq p_i, e_i \leq N, \quad p_i \neq e_i)$, indicating that p_i is directly superior to e_i.

In the next line, there is a non-negative integer Q $\left(0 \leq Q \leq 10^6\right)$.

In the following Q lines, there are pairs a_i, b_i $(1 \leq a_i, b_i \leq N, \quad a_i \neq b_i)$. It is guaranteed that at that moment, a_i will either be directly superior to b_i or vice versa.

In the test data, it is **guaranteed** that it will be possible to achieve at least one hierarchy with some sequence of reversals.

Output

In the next $Q+1$ lines, for each of the given scenarios, it is necessary to output whether the current structure is a hierarchy, i.e., "`DA`" if it is, or "`NE`" if it is not (without quotation marks).

Example

Input	Output
3	DA
1 2	DA
1 3	DA
3	DA
1 2	
1 2	
1 3	
4	DA
2 1	NE
2 3	DA
1 4	DA
4	NE
4 1	
4 1	
3 2	
1 4	

Problem I. Intertwined

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

NCPC (Nordic Cargo Plane Control) are testing a new engine for their cargo planes. To this end they have bound a strong and sturdy infinitely thin rope to the centre of their testing platform, and to the engine. We will place a coordinate system onto this testing platform such that the rope is bound at the origin and lays along the positive x-axis to $(d, 0)$. On this testing platform there are also a number of infinitely thin pillars that can stop the rope, but ignore the engine. As the engine is started it starts rotating the rope counter-clockwise around the origin until it hits a pillar, at which point it is caught and starts rotating around that pillar counter-clockwise instead. The engine is then rotating at a smaller radius as some of the rope is caught between the origin and this pillar. This keeps going until the rope is too short to reach any other pillars.

Running these tests, buying all this infinitely thin rope and setting up these infinitely thin pillars, is expensive. Besides, the workers keep getting these nasty paper-like cuts from all these infinitely thin objects. It would be much more economical to just simulate the behaviour.

Input

The first line contains an integer n, the number of pillars, and an integer d, the length of the rope ($1 \le n \le 10^5$, $1 \le d \le 10^9$).

The following n lines each contain two integers x_i, y_i, the coordinates of the ith pillar ($-10^9 \le x_i, y_i \le 10^9$) for $i \in 1, 2, \ldots, n$. None of these pillars will lie on the rope.

Output

Print one line with an integer i, meaning that the rope will end up spinning around the ith pillar in the input. Note that this index is 1-indexed. If the rope doesn't collide with any pillars $i = -1$. It is guaranteed that changing the input d by at most $\pm 10^{-6}$ will not change i.

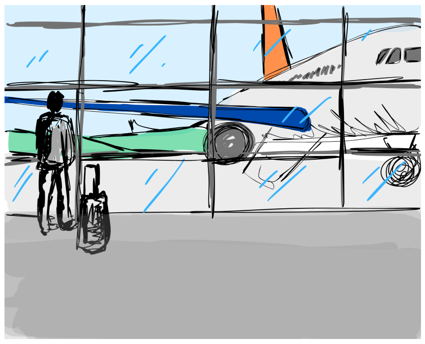
Example

Input	Output

5 200
4 4
4 -4
3 1
-4 4
-4 -4
	1

Problem J. Journey of Recovery

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



You are making an international trip with several stops to blow off steam and celebrate your progression onto the NWERC. Since your flights are often booked with low-cost airlines, you always run the risk of your flights being cancelled last minute leaving you stuck in the airport. Normally this is no problem—take the next flight—but you have to arrive at the NWERC on time.

If any one of your flights is cancelled at the same moment you are about to depart, and all others operate as planned, you will book a new itinerary from there to your final destination. Assuming you always plot the fastest route, by how much will you be delayed in the worst case?

Input

- One line containing the number of flight connections overall, n ($1 \leq n \leq 10^6$).
- n further lines, the ith of which contains four space-separated fields:
  - The code of the departure airport, s_i ($1 \leq |s| \leq 20$)
  - The time of departure in days, minutes, and hours, in the format `ddhh:mm` ($1 \leq d \leq 365$, $0 \leq hh \leq 23$, $0 \leq mm \leq 59$).
  - The code of the arrival airport, t_i ($1 \leq |s| \leq 20$)
  - The time of arrival in days, minutes, and hours, in the format `ddhh:mm` ($1 \leq d \leq 365$, $0 \leq hh \leq 23$, $0 \leq mm \leq 59$).
- One line containing the number of flight connections in your itinerary, m ($1 \leq m \leq n$).
- One line containing the m indices $f_1 \ldots f_m$ of flight connections, in the order you plan to take them.

Flights always go between different airports and always strictly forward in time. For every consecutive pair u, v in your itinerary, the arrival time of flight u is guaranteed to be less than or equal to the departure time of flight v.

Transfers are instantaneous—that is to say, arriving at an airport and departing from it in the same minute is possible. Likewise, if one planned flight is cancelled, you may board another departing at exactly the same time.

Output

Output the maximum amount by which you could be delayed if any one of the given flights is cancelled at its moment of boarding. If you would not be delayed at all in any case (or can even arrive early) simply output 0.

If you cannot always make it to the destination at all, output `stranded` instead.

Example

Input	Output
8 egnx 0d00:10 delft 0d01:00 delft 0d01:00 zad 0d09:00 zad 0d09:01 prg 0d15:30 prg 0d20:00 delft 1d02:15 prg 0d22:00 delft 1d04:15 zad 2d00:00 delft 3d00:00 egnx 2d00:00 delft 2d02:00 egnx 2d00:00 delft 2d02:00 4 1 2 3 4	2745
3 ork 101d00:00 noc 101d00:01 ork 100d23:59 noc 101d00:02 dub 100d00:00 ork 101d00:00 2 3 1	stranded
2 lax 0d00:30 hnl 0d06:20 lax 0d00:30 hnl 0d06:20 1 2	0

Problem K. Ketek Counting

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

Define a *Ketek* to be a sentence that reads the same forwards and backwards, by word. For example, "fall leaves after leaves fall" is a *Ketek* since the words in reverse order are the same as the original order.

Given a string consisting of lower-case letters and the character '`?`', count the number of distinct *Keteks* you can make by replacing every '`?`' with lower-case letters (one letter per '`?`'), and optionally adding spaces between any letters. Note that a *Ketek* cannot contain any `?`'s; they all must be replaced exclusively by lower-case letters.

For example, if we start with the string "`ababa`", we can form 3 different *Keteks*: "`ababa`", "`a bab a`" and "`a b a b a`".

If we start with the string "`?x?z`" instead, we can form 703 different *Keteks*:

- There are $26^2 = 676$ ways to replace the `?`'s and form a one-word *Ketek*.
- Add spaces to form "`? x? z`". There are 26 ways to form a *Ketek* (the first '`?`' must be `z`; the other can be any lower-case letter).
- Add a space to form "`?x ?z`". There is no way to form a *Ketek*.
- Add spaces to form "`? x ? z`". There is one way to form a *Ketek* (the first '`?`' must be `z`; the second must be `x`).

The total is $676 + 26 + 0 + 1 = 703$.

Two *Keteks* are different if they have a different number of words, or there is some word index where the words are not the same.

Input

The single line of input contains a string s ($1 \leq |s| \leq 30000$), which consists of lower-case letters ('`a`' - '`z`') and the character '`?`'.

Output

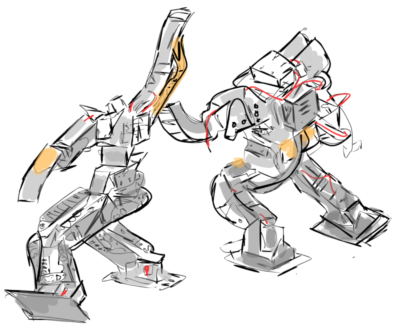
Output the number of distinct *Keteks* that can be formed by replacing the `?`'s with lower-case letters and adding spaces. Since this number may be large, output it modulo 998244353.

Example

Input	Output
ababa	3
?x?z	703

Problem L. Last One Standing

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



In a computer game units are described by their health h, damage d, and time to reload t.

When such a unit fires a missile at an opposing one — the opponent's health is decreased by d 0.5 seconds after the missile is fired. The time between consecutive missile launches for the same unit should be at least t seconds.

For simplicity, we assume the missile supply to be infinite for all units in the game.

Two players — one controlling a unit with health h_1, damage d_1 and time to reload t_1, and the second with a unit described by h_2, d_2 and t_2 — have engaged in a fight in this computer game. Both units are fully reloaded at the beginning of the fight and can fire missiles immediately.

The unit is destroyed when its health becomes zero or negative. A player wins if there is a moment in time such that the opponent's unit is destroyed, while theirs is not.

Since it takes 0.5 seconds for a missile to reach its target, it is possible for both units to fire missiles at the same time and ultimately destroy each other.

You are to determine who wins in case both players act optimally.

Input

- One line containing the integer numbers h_1, d_1 and t_1 ($1 \le h_1, d_1, t_1 \le 1000$).
- One line containing the integer numbers h_2, d_2 and t_2 ($1 \le h_2, d_2, t_2 \le 1000$).

Output

Output the phrase `player one` if the first player wins, `player two` if the second player wins, or `draw` if neither player wins.

Example

Input	Output

`30 10 10`
`30 15 19`
	`player two`

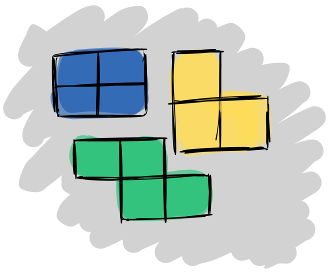
`30 15 19`
`30 10 10`
	`player one`

`100 20 10`
`100 12 5`
	`draw`

Problem M. Mini-Tetris 3023

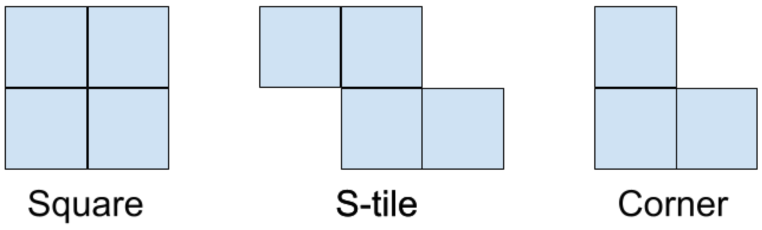
Source file name:	Minitetris.c, Minitetris.cpp, Minitetris.java, Minitetris.py
Input:	`Standard`
Output:	`Standard`

A guy named Gry found a new game called "Mini-Tetris 3023". This small version of Tetris is played on a very long grid only 2 cells high and has just three types of tile:



- A `square` made out of 4 tiles in a 2×2 grid.
- An `S-tile` made out of 4 tiles, 2 on one row and 2 slightly offset on the other
- A `corner` made out of 3 tiles, 1 on one row and 2 on the other

Tiles may be rotated 0, 90, 180, or 270 degrees to fit amongst each other, however, they cannot overlap or go outside the vertical boundary of the grid.



This game provides a squares, b S-tiles, and c corners. Gry would like to beat the high score by creating the largest-possible contiguous $2 \times n$ rectangle out of some or all of the provided tiles, without any tiles overlapping or sticking out of the rectangle.

Input

- The sole line of input contains three integers a, b, and c ($0 \leq a, b, c \leq 50$) — the number of squares, S-tiles, and corners, respectively.

Output

Output the maximum possible width of the grid, n, that can be perfectly filled by some or all of the given tiles without overlapping or overstepping the boundaries.

Example

Input	Output
`2 2 2`	`11`
`1 1 1`	`2`
`0 0 0`	`0`