Problem A. Alien Attack

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

A representative of the Galactic Committee for Person Captures.
By D J Shin on Wikimedia Commons

Aliens are visiting Earth and, as usual, they plan to abduct humans for their experiments. In the past, alien abductions have caused a lot of press coverage and wild speculation on Earth. Luckily for them, most people do not believe these stories and think that aliens are not real.

In order to keep a low profile in the future, the Galactic Committee for Person Captures (GCPC) has established rules for abductions. Besides a lot of boring paperwork, the aliens have to prepare the abduction carefully. While they can make multiple trips (in fact, alien travel is so fast in practice that this is not a limitation at all), they must be smart about it so that their secret is not revealed to humans. If aliens want to abduct a person, they are required to abduct all of their friends at the same time, so that no one notices that their friend is missing when they want to hang out. Of course, friendships on planet Earth are bidirectional, that is if Alice is a friend of Bob, then Bob is also a friend of Alice.

In preparation for the trip, the aliens have observed their targets and started taking note of all their friendships. In total, they must abduct n people, including their friends. Now, they want to book a starship at their local dealership and wonder how much space they need to abduct all n people. A starship's storage space is measured in terms of the number of people that can be transported simultaneously. What is the minimum storage space required to abduct all n people?

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 2 \cdot 10^5$, $0 \leq m \leq 2 \cdot 10^5$), the number of people and the total number of friendships between them.
- m lines, each with two integers i and j ($1 \leq i < j \leq n$), denoting a friendship between persons i and j.

The people are numbered from 1 to n. It is guaranteed that no friendship is listed multiple times.

Output

Output the minimum storage space needed to abduct all people.

Example

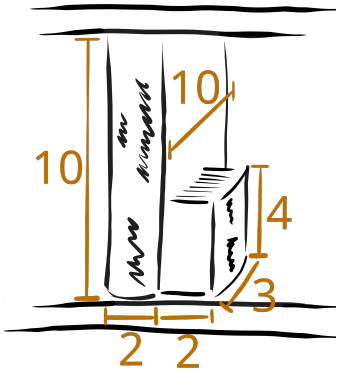
Input	Output
5 3 1 2 2 3 4 5	3
3 0	1
8 8 1 2 2 3 3 4 1 4 1 5 2 6 3 7 4 8	8

Problem B. Bookshelf Bottleneck

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

Brianna is a bookworm. At home, she has a big bookshelf with all her favorite books. She has a large collection ranging from detective novels and science-fiction novels to biographies.

Recently, Brianna has expanded her collection with n graphic novels. However, the new books currently lie around everywhere and form huge stacks on the floor. In the meantime, one of the shelf boards has collected dust and random household utensils that do not belong there. The new books just lying around have become too much to bear, and Brianna finally decided to put them on this shelf board. To do so, she first has to make room on it.



Visualization of Example Input 3.

Brianna wants to arrange the books in a single horizontal line without stacking multiple books on top of each other. While the shelf is wide enough to hold all books without problems, it takes time to make room on the shelf. Therefore, Brianna wants to minimize the width of the part of the shelf that she needs to clear.

Each book can be described as a cuboid with three side lengths l, w, and h. Since the room above the shelf board is limited by the next shelf board above it, she can only fit a book vertically if its vertical side length is at most the distance H between the two shelf boards. Brianna may rotate each book in three-dimensional space as she wants. It is guaranteed that the shelf is deep enough so that the books will not fall off, no matter the orientation. However, all books must stand properly on the shelf board, meaning that every book touches the shelf board along an entire face and not just by an edge.

What is the minimum width of shelf Brianna's books need?

Input

The input consists of:

- One line with two integers n and H ($1 \leq n \leq 10^5$, $1 \leq H \leq 10^9$), the number of books and the height of the shelf, respectively.
- n lines, each containing three integers l, w, h ($1 \leq l, w, h \leq 10^9$), the dimensions of the books.

Output

Output the minimum width of shelf Brianna's books need, or "`impossible`" if it is impossible to place the books on the shelf.

Example

Input	Output
1 3 10 2 5	5
1 3 10 4 5	impossible
2 10 10 2 10 2 3 4	4
3 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000	3000000000

Problem C. Copycat Catcher

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

Your university recently established the Graduate Code Plagiarism Control (GCPC) initiative to get hold of the ever-increasing load on the graders for the computer science assignments. Currently, the graders need to check the code of assignments manually for plagiarism. The GCPC aims to simplify this part of the graders' jobs by performing the plagiarism checks automatically.

A Plagiarism Keyboard

Code consists of tokens separated by spaces. Tokens are strings of alphabetical letters, numerals, and brackets. If a token consists of only a single alphabetical letter (upper or lowercase), it is a variable in the code.

The GCPC wants the plagiarism checker to compare query pieces of code to a reference code. Specifically, it should check whether each query could have been obtained by selecting a contiguous string of tokens from the reference and consistently renaming variables. Variables are consistently renamed if no two occurrences of the same variable are renamed to different variables, and if no two different variables are renamed to the same variable.

The GCPC has asked you to develop the plagiarism checker.

Input

The input consists of:

- A description of the reference, consisting of:
  - One line containing an integer n ($1 \leq n \leq 2\,000$), the number of tokens in the reference.
  - One line containing n tokens, each consisting only of the characters '`a`'-'`z`', '`A`'-'`Z`', '`0`'-'`9`', '`(`' and '`)`'.
- An integer q ($1 \leq q \leq 2\,000$), the number of queries.
- $2 \cdot q$ lines, each two lines in the same format as the reference.

It is guaranteed that each query as well as the reference consist of at most 2 000 characters (excluding spaces). Tokens are separated by single spaces.

Output

For each query, output "`yes`" if the query could have been obtained from the reference, and "`no`" otherwise.

Example

Input	Output
9 for i in range(10) do print i j end 4 3 print j i 2 do print 6 k in range(10) do print k 6 k in range(10) do print j	yes yes yes no
5 i is i times j 7 5 i is i times j 5 a is a times b 5 j is j times c 5 a is i times j 5 j is i times j 5 0 is 0 times j 5 i is i times i	yes yes yes no no no no
5 A 1 ( ) b 4 2 b 2 2 b 1 3 1) (5 a 1 ( ) F	no yes no yes

Problem D. Dark Alley

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

One cold and foggy night, you walk down a shady alley. There should be a lamp every few meters but none of them seem to work, and in this night, not even the moon enlightens your path. Alone and in the dark, you wonder: "Even if there was a working lamp somewhere, how much would it lighten my way?". Now, back at home, you want to calculate this.

A foggy alley. Photo by Henryk Niestrój

The alley can be modeled as a line with a length of n meters. The fog has a uniform density and reduces the light of a lamp by a factor of $1 - p$ every meter. The brightness at one point is the sum of the light that reaches this point from every lamp. You want to calculate this brightness at some points after placing some lamps.

Input

The input consists of:

- One line with two integers n and q and one real number p ($1 \leq n, q \leq 2 \cdot 10^5, 0 < p < 1$), the length of the alley, the number of queries and the density of the fog. The density p of the fog will be given with at most 6 digits behind the decimal point.
- q lines containing one of three query types:
  - "`+ b x`" given two integers b and x ($1 \leq b \leq 10^9$ and $1 \leq x \leq n$), place a lamp with brightness b at position x.
  - "`- b x`" given integers b and x ($1 \leq b \leq 10^9$ and $1 \leq x \leq n$), remove a lamp with brightness b at position x. It is guaranteed that a lamp with that brightness was placed there earlier.
  - "`? x`" given one integer x ($1 \leq x \leq n$), calculate the brightness at position x.

Output

It can be shown that the brightness can be calculated as a fraction $\frac{P}{Q}$ where Q is not divisible by $10^9 + 7$. For each query of type "`?`", print the brightness as $P \cdot Q^{-1} \bmod 10^9 + 7$ in a single line.

Example

Input	Output
5 6 0.25 + 4 2 ? 1 ? 2 ? 3 ? 4 ? 5	3 4 3 250000004 187500003
5 7 0.33 + 9 1 ? 5 + 4 3 ? 2 ? 5 - 9 1 ? 2	312342734 470000012 341542736 760000008

Explanation

For the example 1. The brightness in the alley after placing the lamp will look like this:

3	4	3	2.25	1.6875

Problem E. Eightgon

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

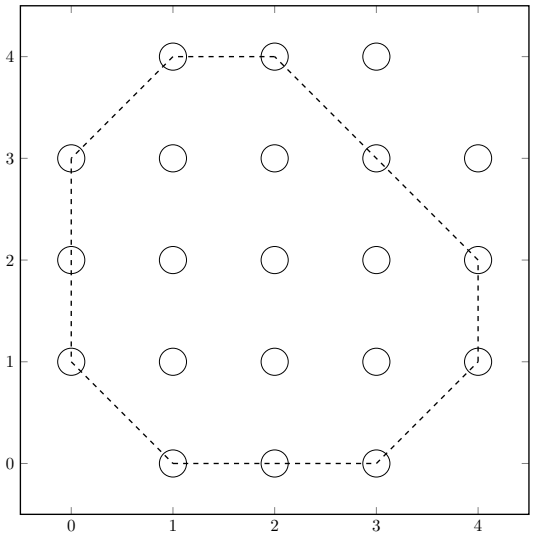
After many years you and your coauthor H. Addaway have finally developed a Theory of Everything that explains everything: Why does time have a direction? How should quantum mechanics be interpreted? What caused the Big Bang? What is love?

An unfortunate fact about physics is that physical theories need to be experimentally tested. In particular, your theory rests on the discovery of so called Barely Audible Particle Clusters (BAPCs). For this purpose you have proposed the development of a Large Eightgon Collider. What remains is to find a suitable location to construct this scientific wonder.

For obvious reasons, the Large Eightgon Collider must consist of eight straight tunnels that together form an underground cycle. Each tunnel is allowed to have a different non-zero length. At each of the eight tunnel connections, a special detector must be built, that also slightly deflects the particles 45 degrees to the left. Each of the eight detectors attracts many researchers, requiring a shaft to the surface to supply them with fresh food and oxygen.

In order to save costs, they will reuse abandoned mine shafts. Given a map of all abandoned mine shafts, your job is to find the number of possible locations to build this miracle. You only consider locations where at least one tunnel runs parallel to the x-axis of the map.

The figure shows the second example.



Visualization of Example 2 showing one possible location for the Large Eightgon Collider.

Input

The input consists of:

- A line with an integer n ($1 \leq n \leq 5000$), the number of abandoned mine shafts.
- n lines, each with two integers x and y ($-10^8 \leq x, y \leq 10^8$), the coordinates of the abandoned mine shafts.

Output

Output the number of possible locations to build the Large Eightgon Collider.

Example

Input	Output
8 0 1 1 0 0 2 2 0 3 1 1 3 3 2 2 3	1
21 0 1 0 2 0 3 1 0 1 1 1 2 1 3 1 4 2 0 2 1 2 2 2 3 2 4 3 0 3 1 3 2 3 3 3 4 4 1 4 2 4 3	15

Problem F. Figure Skating

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

CC-BY-SA 4.0 By Sandro Halank on commons.wikimedia.org

Figure skating is a very popular sport at the Winter Olympics. It has been on the programme the longest of all winter sports, having even been included in the Summer Olympics before the split in 1924. Just like in gymnastics, each contestant executes a routine consisting of elements, which are individually scored by a jury. This subjective aspect to judging skill always leaves room for heated discussion, but a huge scandal in the 2002 Winter Olympics, with allegations that the game had been fixed, caused a transition to the new scoring system IJS. Points awarded to each element of the routine are known beforehand: A Lutz scores 0.60 points (but 2.10 for a double and 5.90 for a triple), a Salchow scores 0.40 (1.30 for double, 4.30 for triple), an Euler scores 0.50, et cetera. Then, points are added or subtracted by the jury based on execution. Consequently, a figure skater is able to estimate his or her score assuming average performance.

Olympics observers from the Bookmakers' Association for the Prevention of Cheating are tasked with assessing the objectivity of the jury. They will compare the predicted ranking of the contestants with the final outcome to determine who is the jury's favourite. The favourite is the contestant who rose the most places between the predicted and final scoreboard. Ties are broken by whoever ends up higher on the final scoreboard. However, if no one did better than predicted, this raises some red flags with the observers, which is declared "suspicious".

Input

The input consists of:

- A line containing a single integer n ($1 \leq n \leq 1000$), the number of contestants.
- n lines, the ith of which contains the name of the contestant who places ith on the predicted scoreboard.
- n lines, the ith of which contains the name of the contestant who places ith on the final scoreboard.

Each name consists of at most 100 lower-case and upper-case alphabetical characters. All names are unique, and occur on both scoreboards exactly once.

Output

If the scoreboards are suspicious, output "`suspicious`". Otherwise, output the name of the jury's favorite.

Example

Input	Output
3 Plisetsky Katsuki Leroy Leroy Plisetsky Katsuki	Leroy
2 Allison Bobson Allison Bobson	suspicious
3 daSilva Aziz Peters Aziz Peters daSilva	Aziz

Problem G. Great Expectations

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

A *speedrun* is a playthrough of a game with the intention to complete it as quickly as possible. When speedrunning, you usually follow a pre-planned path through the game. Along this path, there may be some places where you have to pull off a difficult technique, or *trick*, which may cause a delay if you fail to pull it off successfully. Luckily you can *reset* the game at any time: if you have made a few mistakes, you can start a new run, losing your progress but instantaneously starting over with a clean slate. You can do this as often as you like.

The game you are currently speedrunning has a record of r seconds, which you intend to beat. You have discovered a path through the game that, in the best case, takes $n < r$ seconds. There are some tricks along the way, though: you know exactly where along the run they occur, what the probability is that you will pull them off successfully, and how many seconds you have to spend to recover if they fail.

Given this data, you want to find the optimal strategy for when to reset the game to minimise the expected time to set a new record. Write a program to determine what this smallest possible expected time is.

Input

The input consists of:

- One line with three integers n, r and m ($2 \leq n < r \leq 5\,000$, $1 \leq m \leq 50$), where n and r are as described above and m is the number of tricks.
- m lines, each containing three numbers describing a trick:
  - An integer t ($1 \leq t < n$), the time in the route (assuming no failed tricks before) at which the trick occurs,
  - a real number p ($0 < p < 1$ and p has at most 6 digits after the decimal point), the probability that the trick succeeds, and
  - an integer d ($1 \leq d \leq 1\,000$), the number of seconds required to recover in case the trick fails.

The tricks are given in sorted order by t, and no two tricks occur at the same time t in the route.

You may assume that, without resetting, a single playthrough has a probability of at least 1 in 50 000 to succeed at improving the record.

Output

Output the expected time you will have to play the game to set a new record, assuming an optimal strategy is used. Your answer should have an absolute or relative error of at most 10^{-6}.

Example

Input	Output
100 111 5 20 0.5 10 80 0.5 2 85 0.5 2 90 0.5 2 95 0.5 2	124
2 4 1 1 0.5 5	3
10 20 3 5 0.3 8 6 0.8 3 8 0.9 3	18.9029850746
10 50 1 5 0.5 30	15

Explanation

Example Input 1:

The record for this game is 111 seconds, and your route takes 100 seconds if everything goes right.

After playing for 20 seconds, there is a trick with a 50% success rate. If it succeeds, you keep playing. If it fails, you incur a 10 second time loss: now the run will take at least 110 seconds. It is still possible to set a record, but every other trick in the run has to be successful. It turns out to be faster on average to reset after failing the first trick.

Thus you repeat the first 20 seconds of the game until the trick is successful: with probability 1/2, it takes 1 attempt; with probability 1/4, it takes 2 attempts; and so on. On average, you spend 40 seconds on the first 20 seconds of the route.

Once you have successfully performed the first trick, you want to finish the run no matter the result of the other tricks: it takes 80 seconds, plus on average 1 second loss from each of the remaining 4 tricks. So the expected time until you set a record is 124 seconds.

Problem H. Headline Heat

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

The German ICPC scene is widely considered one of the most competitive. At least, that's what we tell our students. Countless rivalries form a complex web of envy, despair, glory, and triumph woven around and manifested in the unrelenting echoes of two scoreboards – Winter Contest and GCPC. While generations of participants tend to forget the grudges of their predecessors, we coaches, acting as timeless beacons of continuity, preserve these petty conflicts between long forgotten teams. Striving for perfect balance, we express our dedication to our coaching duties in a furious outcry on social media against every unfair news article. That is, a coach gets mad if a news article is published that contains a rival university name more often than their own.

RANK	TEAM		SCORE	
1	Participants	Kindergarten Timelimit Karlsruher Institut für Technologie	13	1041
2		Seems to be O(kl) Hasso-Plattner-Institut	13	1234
3		<(OvO)> Universität des Saarlandes	12	1080
4		Infinite Loopers Karlsruher Institut für Technologie	12	1178
5		Don't Starve TUMgether Technische Universität München	11	952
6		cracK IT Karlsruher Institut für Technologie	11	1450
7		destructor college Constructor University Bremen	10	751
8		r/wth Rheinisch-Westfälische Technische Hochschule Aachen	10	912
9		Burnoutverbot Hasso-Plattner-Institut	10	1041
10		ÉMAÉ Constructor University Bremen	10	1249

Top 10 scoreboard of Winter Contest 2024.

To smoothen the waves of conflict in this ocean of rage, a newly appointed authority is tasked with proofreading media coverage of GCPC and Winter Contest to prevent uneven coverage.

Input

The input consists of:

- One line with the number of universities n, rivalries m, and articles k. ($1 \leq n, m, k \leq 10^5$)
- n lines containing the name of a university p_i.
- m lines containing two integers u, v, meaning that universities u and v are rivals. ($1 \leq u, v \leq n, u \neq v$)
- k lines containing a news article t_i.

If a university u is a rival of university v, then v is also a rival of u. Moreover, there are no duplicate rivalries.

Names and articles are strings of lowercase Latin letters and spaces. The first and last character of a name or article are never a space. Names can overlap and be contained in other names.

The summed length of all names and articles is at most 10^6, i.e. $\sum_{i=1}^{n} |p_i| + \sum_{i=1}^{k} |t_i| \leq 10^6$.

Output

For each article, output "`no`" if it will draw the wrath of at least one coach and "`yes`" otherwise.

Example

Input
3 1 4 hpi fau kit 1 3 kit destroys hpi at wintercontest gcpc is great team moshpit from hpi beats kit teams whats the abbreviation for university of erlangen nuremberg
Output
yes yes no yes

Input
6 3 5 uds cu tum rwth uni ulm uni 4 1 2 5 1 3 last gcpc rwth had a team in top ten two places behind tum who is team debuilding from constructor university bremen top ten teams last year are from kit cu uds hpi tum and rwth uni ulm cu uni ulm sunday alright lets go
Output
no yes no no yes

Problem I. Interference

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

Physics can be so much fun! Yesterday, your teacher explained how interference works: If you have two waves, their heights add up over the whole waves' length! So if both waves have a peak, the resulting peak will be even higher. Likewise, if both waves have a wave trough below the water surface, the resulting wave has a trough that will be even further below. Technically, a wave's height is called amplitude and the distance between two wave peaks is called wavelength.

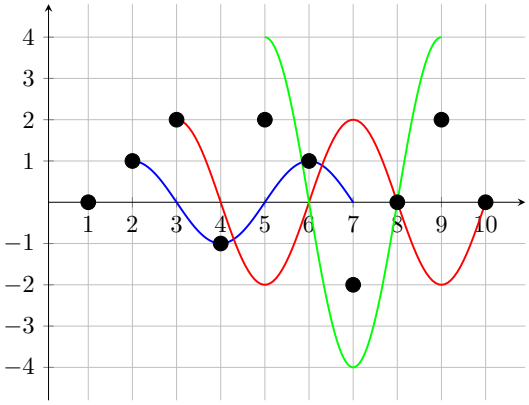


Figure I.1: Interference of three waves in Example Input 2. The black dots represent the resulting wave's height.

Today, your physics teacher describes the setup of an experiment she is about to perform. She will create stationary waves in a one-dimensional container of water. Due to her superior control over physical elements, all waves will have a precisely controlled amplitude and will only be created in an interval of given length. The wavelength of each wave is always 4 and the first positive peak will always be at the first index of the interval. We only measure the wave's amplitude at integer points. For example, a wave with amplitude 2 and length 9 can be described as `2 0 -2 0 2 0 -2 0 2`. If there is no wave at a point, the amplitude is 0. Your task is to predict how high the resulting wave will be at given points in the container taking into account all the waves that were created up to that point.

Input

The input consists of:

- One line with two integers n and w ($1 \leq n \leq 4\,000$, $1 \leq w \leq 10^9$), the number of lines and the width of the container.
- n lines, each containing either a wave description or a prediction task:
  - "`!` p ℓ a", a wave description with starting position p, length ℓ ($1 \leq p, \ell \leq w$), and amplitude a ($1 \leq a \leq 10^9$). It is guaranteed that $p + \ell - 1 \leq w$.
  - "`?` p", a prediction task for the resulting wave at position p ($1 \leq p \leq w$).

See Figure I.1 for a partial visualization of Example 2.

Output

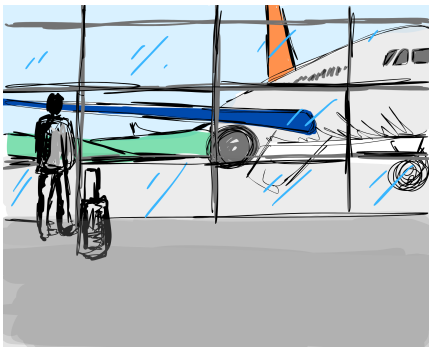
For each prediction task, output a line with a single integer, the height of the wave resulting from all former described waves at the requested position.

Example

Input	Output
4 10 ! 2 7 1 ? 9 ? 7 ? 6	0 0 1
7 10 ! 2 6 1 ! 3 8 2 ! 5 2 3 ? 6 ! 5 5 4 ? 8 ? 9	1 0 2
6 12 ! 1 7 1 ! 7 3 2 ? 6 ? 7 ? 8 ? 10	0 1 0 0
6 11 ! 1 6 1 ? 6 ! 5 7 4 ? 6 ! 6 3 2 ? 6	0 0 2

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

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

Karen has a beautiful precious glass object on the shelf in her living room. Unfortunately, her cat Klaus does not like it when there is stuff on his favourite shelf. Everything that is not bolted or glued in place, he will gradually push over the edge while looking Karen straight in the eyes.

Klaus, 10 seconds before an event that was luckily covered by Karen's insurance.

Now, Klaus' paw slowly executes his diabolical deed. His cute fluffy face radiates inadvertent innocence. Knowing that any intervention would only delay the inevitable, Karen wonders what will happen to the string her sister Kim wrote on the precious glass object. After all, it took Kim a whole week to gather all the `bdpq` letters that make up the string.

Can you describe to Karen what the string will look like from her point of view while it tumbles towards destruction?

While falling off the shelf, Karen's precious glass object is subject to the following transformations, described as seen when looking at the object from the front.

- `h`: horizontal flip, e.g. `bbq` becomes `pdd`
- `v`: vertical flip, e.g. `bbq` becomes `ppd`
- `r`: 180-degree rotation, e.g. `bbq` becomes `bqq`

No flips along or rotations about any other axes are possible.

Input

The input consists of:

- One line with a string s consisting of the letters `bdpq` ($1 \leq |s| \leq 5 \cdot 10^5$), the string printed on the glass object as seen at the start of the fall.
- One line with a string t consisting of the letters `hvr` ($1 \leq |t| \leq 5 \cdot 10^5$) giving the sequence of transformations in the order that they occur during the fall.

Output

Output the string that can be seen at the moment the glass object touches the ground and just before it shatters into pieces.

Example

Input	Output
bbq h	pdd
bbq v	ppd
bbq r	bqq
ppbddbq hvrhv	bqppqdd

Problem L. Laundry

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

Laundry hanging to dry
Image by gregroose on Pixabay

Every Sunday is laundry day, and there is always a huge pile of clothes waiting to be washed, which is certainly going to take you forever. You are particularly annoyed by how careful you have to be when washing certain items, and how important it is that you choose an appropriate washing programme for each item.

Fortunately, your washing machine is quite old and only supports three different washing programmes: A, B, and C. You can put at most k items in one load, and each load can be washed using one of the programmes.

Some items are easy to care for, and you can put them in any load you like. More delicate items must not be washed using a specific programme, but the other two are fine. Of course, the worst clothes are the ones for which only one programme is appropriate.

You have already sorted the items into seven piles by putting items together for which the same combination of programmes is fine, so you know how many items are in each pile.

What is the minimum number of loads you need to wash?

Illustration of Example Input 2 with an optimal solution. The figure on the left shows seven piles, one for each combination. The figure on the right shows a (possible) optimal solution, where each pile is washed in one load. The numbers on the pile represent how many items of each combination are washed with this load. In particular, the leftmost pile is washed using programme A, the two piles in the middle with programme B, and the two piles on the right with programme C. Thus, we need five loads to wash all items, which is optimal since we have 15 items in total.

Input

The input starts with a line containing one integer t ($1 \leq t \leq 10^4$), the number of test cases. Then for each test case:

- One line with an integer k ($1 \leq k \leq 10^9$), the number of items you can put in one load.
- One line with seven integers $c_1, \ldots, c_7$ ($0 \leq c_i \leq 10^9$), the number of items for each combination of programmes. The integers are given in this order: A, B, C, AB, BC, AC, ABC. For example, c_4

must be washed using either programme A or programme B.

Output

For each test case, output the minimum number of loads that are needed to wash all clothes.

Example

Input
4 10 15 11 9 5 2 7 1 120 0 0 0 0 0 0 0 6 5 6 8 9 1 0 0 1213 295053681 137950336 87466375 956271897 344992260 31402049 988259763
Output
6 0 6 2342454
Input
1 3 1 2 1 3 3 2 3
Output
5

Problem M. Musical Mending

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

Shortly before the concert starts, you notice that your piano is completely out of tune! Having the ability of relative pitch, you are able to discern the difference between the pitch of any piano key to the first piano key. While this does not help you find the absolute pitch, you decide to at least tune the keys relative to each other. To do this, you need to make sure that the pitch of each key is exactly one higher than the key before it and one lower than the key after it. As the concert will start shortly, you need to minimize the total tuning effort, which is the sum of the absolute changes in pitch you apply to each key. For example, Figure M.1 illustrates a solution for Example Input 3, resulting in a total tuning effort of 23.

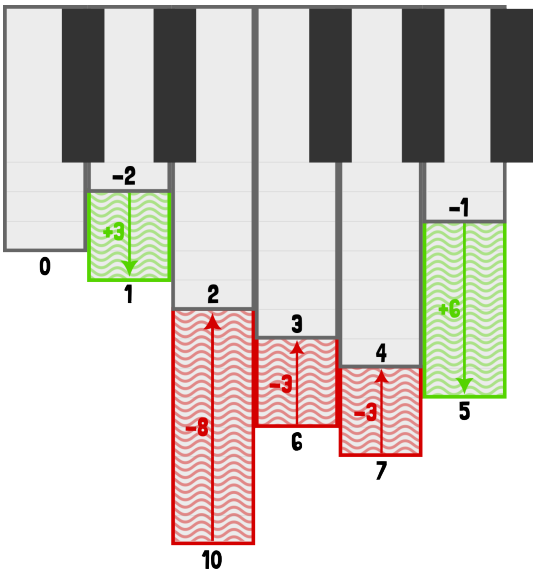


Figure M.1: Visualization of Example Input 3.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 10^5$), the number of keys on the piano.
- One line with n integers $t_1, \ldots, t_n$ ($-2 \cdot 10^5 \leq t_i \leq 2 \cdot 10^5$), where t_i describes the difference in pitch between the ith key and the first key. The first integer t_1 is always 0.

Output

Output the minimal total tuning effort.

Example

Input
7 0 1 2 3 4 5 6
Output
0

Input
5 0 1 4 3 6
Output
4

Input
6 0 -2 10 6 7 -1
Output
23

Input
4 0 -4 -2 1
Output
7

Input
9 0 23452 145043 -3423 -20 9845 435 -3 4453
Output
186237