Problem A
Balanced Diet
Time limit: 2 seconds

Every day, Danny buys one sweet from the candy store and eats it. The store has m types of sweets, numbered from 1 to m. Danny knows that a balanced diet is important and is applying this concept to his sweet purchasing. To each sweet type i, he has assigned a *target fraction*, which is a real number f_i $(0 < f_i \le 1)$. He wants the fraction of sweets of type i among all sweets he has eaten to be roughly equal to f_i. To be more precise, let s_i denote the number of sweets of type i that Danny has eaten, and let $n = \sum_{i=1}^{m} s_i$. We say the set of sweets is *balanced* if for every i we have

$$nf_i - 1 < s_i < nf_i + 1.$$

Danny has been buying and eating sweets for a while and during this entire time the set of sweets has been balanced. He is now wondering how many more sweets he can buy while still fulfilling this condition. Given the target fractions f_i and the sequence of sweets he has eaten so far, determine how many more sweets he can buy and eat so that at any time the set of sweets is balanced.

Input

The input consists of three lines. The first line contains two integers m $(1 \le m \le 10^5)$, which is the number of types of sweets, and k $(0 \le k \le 10^5)$, which is the number of sweets Danny has already eaten.

The second line contains m positive integers $a_1, \ldots, a_m$. These numbers are proportional to $f_1, \ldots, f_m$, that is, $f_i = \frac{a_i}{\sum_{j=1}^{m} a_j}$. It is guaranteed that the sum of all a_j is no larger than 10^5.

The third line contains k integers $b_1, \ldots, b_k$ $(1 \le b_i \le m)$, where each b_i denotes the type of sweet Danny bought and ate on the i^{th} day. It is guaranteed that every prefix of this sequence (including the whole sequence) is balanced.

Output

Display the maximum number of additional sweets that Danny can buy and eat while keeping his diet continuously balanced. If there is no upper limit on the number of sweets, display the word `forever`.

Sample Input 1

```
6 5
2 1 6 3 5 3
1 2 5 3 5
```

Sample Output 1

```
1
```

Sample Input 2

```
6 4
2 1 6 3 5 3
1 2 5 3
```

Sample Output 2

```
forever
```

Problem B
Branch Assignment
Time limit: 3 seconds

The Innovative Consumer Products Company (ICPC) is planning to start a top-secret project. This project consists of s subprojects. There will be $b \geq s$ branches of ICPC involved in this project and ICPC wants to assign each branch to one of the subprojects. In other words, the branches will form s disjoint groups, with each group in charge of a subproject.

At the end of each month, each branch will send a message to every other branch in its group (a different message to each branch). ICPC has a particular protocol for its communications. Each branch i has a secret key k_i known only to the branch and the ICPC headquarters. Assume branch i wants to send a message to branch j. Branch i encrypts its message with its key k_i. A trusted courier picks up this message from this branch and delivers it to the ICPC headquarters. Headquarters decrypts the message with key k_i and re-encrypts it with key k_j. The courier then delivers this newly encrypted message to branch j, which decrypts it with its own key k_j. For security reasons, a courier can carry only one message at a time.

Given a road network and the locations of branches and the headquarters in this network, your task is to determine the minimum total distance that the couriers will need to travel to deliver all the end-of-month messages, over all possible assignments of branches to subprojects.

Input

The first line of input contains four integers n, b, s, and r, where n ($2 \leq n \leq 5\,000$) is the number of intersections, b ($1 \leq b \leq n-1$) is the number of branches, s ($1 \leq s \leq b$) is the number of subprojects, and r ($1 \leq r \leq 50\,000$) is the number of roads. The intersections are numbered from 1 through n. The branches are at intersections 1 through b, and the headquarters is at intersection $b+1$. Each of the next r lines contains three integers u, v, and ℓ, indicating a one-way road from intersection u to a different intersection v ($1 \leq u, v \leq n$) of length ℓ ($0 \leq \ell \leq 10\,000$). No ordered pair (u, v) appears more than once, and from any intersection it is possible to reach every other intersection.

Output

Display the minimum total distance that the couriers will need to travel.

Sample Input 1

```
5 4 2 10
5 2 1
2 5 1
3 5 5
4 5 0
1 5 1
2 3 1
3 2 5
2 4 5
2 1 1
3 4 2
```

Sample Output 1

```
13
```

Sample Input 2

```
5 4 2 10
5 2 1
2 5 1
3 5 5
4 5 10
1 5 1
2 3 1
3 2 5
2 4 5
2 1 1
3 4 2
```

Sample Output 2

```
24
```

Problem C
Ceiling Function
Time limit: 5 seconds

Advanced Ceiling Manufacturers (ACM) is analyzing the properties of its new series of Incredibly Collapse-Proof Ceilings (ICPCs). An ICPC consists of n layers of material, each with a different value of collapse resistance (measured as a positive integer). The analysis ACM wants to run will take the collapse-resistance values of the layers, store them in a binary search tree, and check whether the shape of this tree in any way correlates with the quality of the whole construction. Because, well, why should it not?

To be precise, ACM takes the collapse-resistance values for the layers, ordered from the top layer to the bottom layer, and inserts them one-by-one into a tree. The rules for inserting a value v are:

- If the tree is empty, make v the root of the tree.
- If the tree is not empty, compare v with the root of the tree. If v is smaller, insert v into the left subtree of the root, otherwise insert v into the right subtree.

ACM has a set of ceiling prototypes it wants to analyze by trying to collapse them. It wants to take each group of ceiling prototypes that have trees of the same shape and analyze them together.

For example, assume ACM is considering five ceiling prototypes with three layers each, as described by Sample Input 1 and shown in Figure C.1. Notice that the first prototype's top layer has collapse-resistance value 2, the middle layer has value 7, and the bottom layer has value 1. The second prototype has layers with collapse-resistance values of 3, 1, and 4 – and yet these two prototypes induce the same tree shape, so ACM will analyze them together.

Given a set of prototypes, your task is to determine how many different tree shapes they induce.

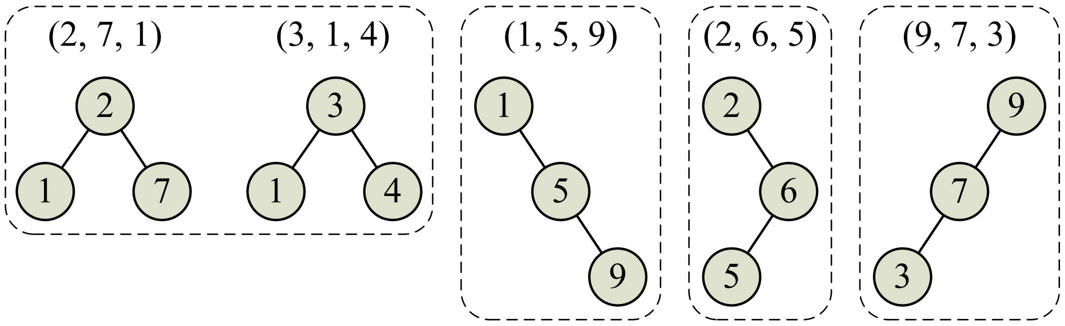

Figure C.1: The four tree shapes induced by the ceiling prototypes in Sample Input 1.

Input

The first line of the input contains two integers n ($1 \leq n \leq 50$), which is the number of ceiling prototypes to analyze, and k ($1 \leq k \leq 20$), which is the number of layers in each of the prototypes.

The next n lines describe the ceiling prototypes. Each of these lines contains k distinct integers (between 1 and 10^6, inclusive), which are the collapse-resistance values of the layers in a ceiling prototype, ordered from top to bottom.

Output

Display the number of different tree shapes.

Sample Input 1

```
5 3
2 7 1
3 1 4
1 5 9
2 6 5
9 7 3
```

Sample Output 1

```
4
```

Sample Input 2

```
3 4
3 1 2 40000
3 4 2 1
33 42 17 23
```

Sample Output 2

```
2
```

Problem D
Clock Breaking
Time limit: 5 seconds

After numerous unfortunate freak fatalities and the lawsuits, settlements, protests, and boycotts that naturally followed, the beleaguered executives at ACME Clock Manufacturers have decided they need to finally fix their disastrous quality control issues. It has been known for years that the digital clocks they manufacture have an unacceptably high ratio of faulty liquid-crystal display (LCD) screens, and yet these heartless souls have repeatedly failed to address the issue, or even warn their hapless consumers!

You have been called in as a quality consultant to finally put a stop to the madness. Your job is to write an automated program that can test a clock and find faults in its display.

These clocks use a standard 7-segment LCD display for all digits (shown on the left in Figure D.1), plus two small segments for the ':', and show all times in a 24-hour format. The minute before midnight is 23:59, and midnight is 0:00. The ':' segments of a working clock are on at all times. The representation of each digit using the seven segments is shown on the right in Figure D.1.

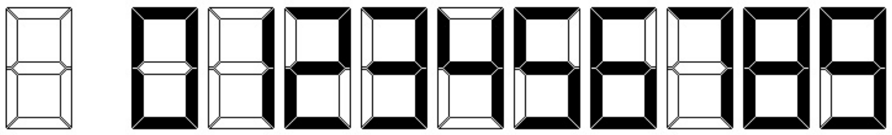

Figure D.1: LCD display of each digit.

Your program will be given the display of a clock at several consecutive minutes, although you do not know exactly what time these displays start. Some of the LCD segments are burnt out (permanently off) and some are burnt in (permanently on). Your program must determine, where possible, which segments are definitely malfunctioning and which are definitely in working order.

Input

The first input line contains a single integer n ($1 \leq n \leq 100$), which is the number of consecutive minutes of a clock's display. The next $8n - 1$ lines contain n ASCII images of these clock displays of size 7×21, with a single blank line separating the representations.

All digit segments are represented by two characters, and each colon segment is represented by one character. The character '`X`' indicates a segment that is on. The character '`.`' indicates anything else (segments that are off or non-segment portions of the display). See the sample input/output for details; the first output shows every possible LCD segment along with the smaller segments used to represent the ':'. No clock representation has an '`X`' in a non-segment position or only half of a segment showing.

Output

Display a 7×21 ASCII image with a '`0`' for every segment that is burnt out, a '`1`' for every segment that is burnt in, a '`W`' for every segment that is definitely working, and a '`?`' for every segment for which the status cannot be determined. Use '`.`' for non-segments. If the given displays cannot come from consecutive minutes, display `impossible`.

Sample Input 1

```
3
......XX.....XX...XX.
.....X..X...X..X....X
.....X..X.X.X..X....X
.............XX...XX.
.....X..X......X.X..X
.....X..X......X.X..X
......XX.....XX...XX.

......XX.....XX...XX.
.....X..X...X..X....X
.....X..X.X.X..X....X
.............XX...XX.
.....X..X......X.X..X
.....X..X......X.X..X
......XX.....XX...XX.

.............XX...XX.
........X...X..X....X
........X.X.X..X....X
.............XX......
........X...X..X.X..X
........X...X..X.X..X
......XX.....XX...XX.
```

Sample Output 1

```
.??...WW.....??...??.
?..?.W..?...?..1.0..?
?..?.W..?.?.?..1.0..?
.??...??.....11...WW.
?..?.W..?.0.W..?.1..?
?..?.W..?...W..?.1..?
.??...11.....??...??.
```

Sample Input 2

```
2
......XX.....XX...XX.
...X....X...X..X.X..X
...X....X.X.X..X.X..X
......XX..........XX.
...X.X....X.X..X.X..X
...X.X......X..X.X..X
......XX.....XX...XX.

......XX.....XX......
...X....X...X..X.....
...X....X.X.X..X.....
......XX.............
...X.X....X.X..X.....
...X.X......X..X.....
......XX.....XX......
```

Sample Output 2

```
impossible
```

Problem E
Forever Young
Time limit: 1 second

My birthday is coming up. Alas, I am getting old and would like to feel young again. Fortunately, I have come up with an excellent way of feeling younger: if I write my age as a number in an appropriately chosen base b, then it appears to be smaller. For instance, suppose my age in base 10 is 32. Written in base 16 it is only 20!

However, I cannot choose an arbitrary base when doing this. If my age written in base b contains digits other than 0 to 9, then it will be obvious that I am cheating, which defeats the purpose. In addition, if my age written in base b is too small then it would again be obvious that I am cheating.

Given my age y and a lower bound ℓ on how small I want my age to appear, find the largest base b such that y written in base b contains only decimal digits, and is at least ℓ when interpreted as a number in base 10.

Input

The input consists of a single line containing two base 10 integers y ($10 \le y \le 10^{18}$ – yes, I am *very* old) and ℓ ($10 \le \ell \le y$).

Output

Display the largest base b as described above.

Sample Input 1	Sample Output 1
`32 20`	`16`

Sample Input 2	Sample Output 2
`2016 100`	`42`

Problem F
Longest Rivers
Time limit: 10 seconds

The Chao Phraya River System is the main river system of Thailand. Its six longest rivers listed by decreasing length are:

1. Tha Chin (765 km)
2. Nan (740 km)
3. Yom (700 km)
4. Ping (658 km)
5. Pa Sak (513 km)
6. Wang (335 km)

The Chao Phraya River System. Picture from Wikimedia Commons.

A simplified model of this river system is shown in Figure F.1, where the smaller red numbers indicate the lengths of various sections of each river. The point where two or more rivers meet as they flow downstream is called a *confluence*. Confluences are labeled with the larger black numbers. In this model, each river either ends at a confluence or flows into the sea, which is labeled with the special confluence number 0. When two or more rivers meet at a confluence (other than confluence 0), the resulting merged river takes the name of one of those rivers. For example, the Ping and the Wang meet at confluence 1 with the resulting merged river retaining the name Ping. With this naming, the Ping has length 658 km while the Wang is only 335 km. If instead the merged river had been named Wang, then the length of the Wang would be 688 km while the length of the Ping would be only 305 km.

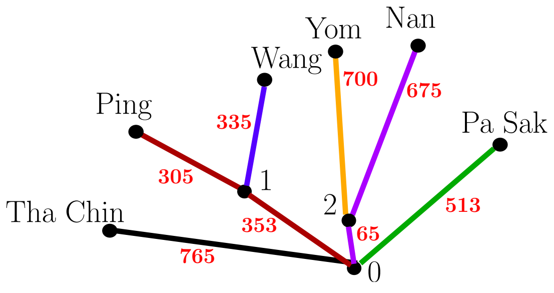

Figure F.1: The river system in Sample Input 1. Same-colored edges indicate a river.

The raised awareness of this phenomenon causes bitter rivalries among the towns along the rivers. For example, the townspeople along the Wang protest that maybe with a proper naming scheme, their river could actually be the longest, or maybe the second longest (or at least not last!). To end all the guessing, your task is to validate all such claims.

The *rank* of a river is its position in a list of all rivers ordered by decreasing length, where the best rank is 1 for the longest river. For each river, determine its best possible rank over all naming schemes. At any confluence, the name of a new, larger river in any naming scheme must be one of the names of the smaller rivers which join at that confluence. If two or more rivers have equal lengths in a naming scheme, all the tied rivers are considered to have the best possible ranking. For example, if one river is the longest and all other rivers are equal, those rivers all have rank 2.

Input

The first line of input contains two integers n $(1 \leq n \leq 500\,000)$, which is the number of river sources in the system, and m $(0 \leq m \leq n-1)$, which is the number of confluences with positive labels. These confluences are numbered from 1 to m.

The next n lines describe the rivers. Each of these lines consists of a string, which is the name of the river at the source, and two integers c $(0 \leq c \leq m)$ and d $(1 \leq d \leq 10^9)$, where c is the identifier of the nearest confluence downstream, and d is the distance from the source to that confluence in kilometers. River names use only lowercase and uppercase letters a–z, and consist of between 1 and 10 characters, inclusive.

The final m lines describe confluences 1 to m in a similar fashion. The k^{th} of these lines describes the confluence with identifier k and contains two integers: the identifier of the nearest confluence downstream and the distance from confluence k to this confluence in kilometers.

It is guaranteed that each confluence 1 through m appears as "the nearest downstream" at least twice, confluence 0 appears at least once, and all sources are connected to confluence 0.

Output

Display one river per line in the same order as in the input. On that line, display the name of the river and its best possible rank.

Sample Input 1

```
6 2
PaSak 0 513
Nan 2 675
Yom 2 700
Wang 1 335
Ping 1 305
ThaChin 0 765
0 353
0 65
```

Sample Output 1

```
PaSak 5
Nan 2
Yom 1
Wang 3
Ping 4
ThaChin 1
```

Problem G
Oil
Time limit: 10 seconds

A large part of the world economy depends on oil, which is why research into new methods for finding and extracting oil is still active. Profits of oil companies depend in part on how efficiently they can drill for oil. The International Crude Petroleum Consortium (ICPC) hopes that extensive computer simulations will make it easier to determine how to drill oil wells in the best possible way.

Drilling oil wells optimally is getting harder each day – the newly discovered oil deposits often do not form a single body, but are split into many parts. The ICPC is currently concerned with stratified deposits, as illustrated in Figure G.1.

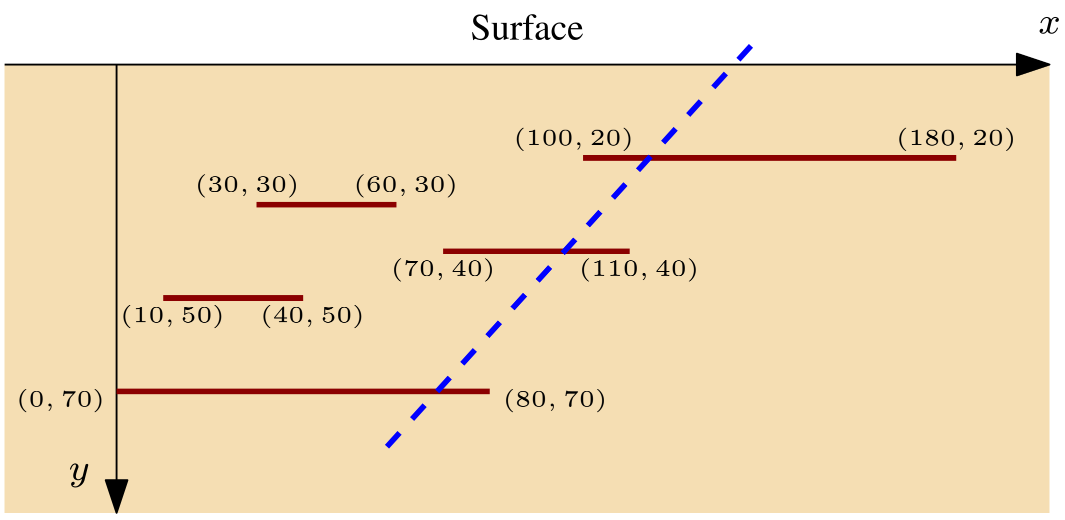

Figure G.1: Oil layers buried in the earth. This figure corresponds to Sample Input 1.

To simplify its analysis, the ICPC considers only the 2-dimensional case, where oil deposits are modeled as horizontal line segments parallel to the earth's surface. The ICPC wants to know how to place a single oil well to extract the maximum amount of oil. The oil well is drilled from the surface along a straight line and can extract oil from all deposits that it intersects on its way down, even if the intersection is at an endpoint of a deposit. One such well is shown as a dashed line in Figure G.1, hitting three deposits. In this simple model the amount of oil contained in a deposit is equal to the width of the deposit. Can you help the ICPC determine the maximum amount of oil that can be extracted by a single well?

Input

The first line of input contains a single integer n ($1 \le n \le 2\,000$), which is the number of oil deposits. This is followed by n lines, each describing a single deposit. These lines contain three integers x_0, x_1, and y giving the deposit's position as the line segment with endpoints (x_0, y) and (x_1, y). These numbers satisfy $|x_0|, |x_1| \le 10^6$ and $1 \le y \le 10^6$. No two deposits will intersect, not even at a point.

Output

Display the maximum amount of oil that can be extracted by a single oil well.

Sample Input 1

```
5
100 180 20
30 60 30
70 110 40
10 40 50
0 80 70
```

Sample Output 1

```
200
```

Sample Input 2

```
3
50 60 10
-42 -42 20
25 0 10
```

Sample Output 2

```
25
```

Problem H
Polygonal Puzzle
Time limit: 20 seconds

During last year's ACM ICPC World Finals in Marrakesh, one of the judges bought a pretty wooden puzzle depicting a camel and palm trees (see Figure H.1). Unlike traditional jigsaw puzzles, which are usually created by cutting up an existing rectangular picture, all the pieces of this puzzle have been cut and painted separately. As a result, adjacent pieces often do not share common picture elements or colors. Moreover, the resulting picture itself is irregularly shaped. Given these properties, the shape of individual pieces is often the only possible way to tell where each piece should be placed.

Figure H.1: The judge's wooden puzzle.

The judge has been wondering ever since last year whether it is possible to write a program to solve this puzzle. An important part of such a program is a method to evaluate how well two puzzle pieces "match" each other. The better the match, the more likely it is that those pieces are adjacent in the puzzle.

Pieces are modeled as simple polygons. Your task is to find a placement of two given polygons such that their interiors do not overlap but the polygons touch with their boundaries and the length of the common boundary is maximized. For this placement, polygons can be translated and rotated, but not reflected or resized. Figure H.2 illustrates the optimal placement for Sample Input 1.

Input

The input contains the description of two polygons, one after the other. Each polygon description starts with a line containing an integer n ($3 \leq n \leq 50$) denoting the number of vertices of the polygon. This is followed by n lines, each containing two integer coordinates x, y of a polygon vertex ($|x|, |y| \leq 100$). The vertices of each polygon are given in clockwise order, and no three consecutive vertices are collinear.

The input data is chosen so that even if the vertices were moved by a distance of up to 10^{-7}, the answer would not increase by more than 10^{-4}.

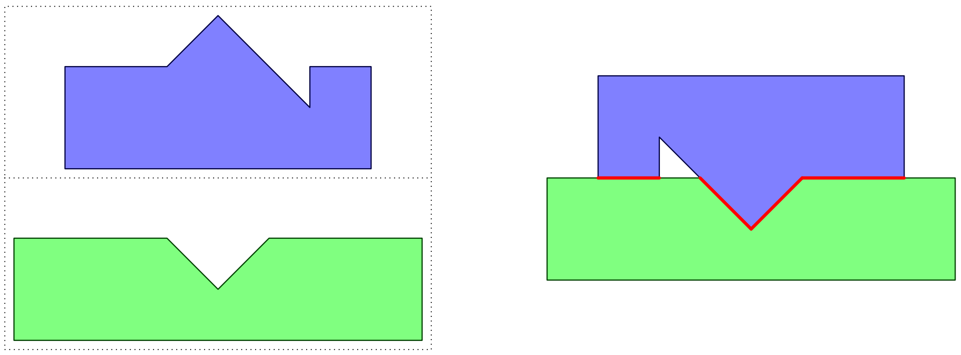

Figure H.2: Sample Input 1 and its optimal placement.

Output

Display the maximum possible length of the common boundary of these polygons when they are optimally placed. Your answer should have an absolute or relative error of less than 10^{-3}.

Sample Input 1

```
8
0 0
0 10
10 10
15 15
24 6
24 10
30 10
30 0
7
-5 0
-5 10
10 10
15 5
20 10
35 10
35 0
```

Sample Output 1

```
30.142135624
```

Sample Input 2

```
3
1 0
0 30
40 0
3
1 0
0 30
40 0
```

Sample Output 2

```
50
```

Problem I
Road Times
Time limit: 5 seconds

Ubol Narongdid is the founder of a brash new startup company called Special D-Liver-E. She wants to corner the market on overnight deliveries of organs between hospitals in the Phuket area. For scheduling purposes it is important to have accurate estimates for the times to perform such deliveries. Several trips between various hospitals have already been performed, so delivery times between those pairs of hospitals are known. The company currently has software to estimate times for other (as yet untraveled) trips, but so far all the estimates have been woefully inaccurate.

You have been asked to come up with a method to improve these estimates. You have at your disposal the following information: 1) the length (in kilometers) of the roads connecting each pair of cities in the Phuket area, and 2) a set of times (in minutes) for various previously executed deliveries.

You know that roads are one-way, and each road has a fixed speed limit that lies between 30 and 60 kilometers per hour. Speed limits are real-valued and need not be integers. You also know that delivery trucks always take the route that minimizes distance traveled, and on each road will always travel at a constant speed equal to that road's speed limit. Thus you know, for example, that if a given trip is 50 kilometers, the time it will take is between 50 and 100 minutes inclusive, in the absence of any other information. Ah, but you *do* have other information, namely the times of previous deliveries. It is up to you to use it to produce the best possible estimates.

Input

The input starts with a line containing an integer n ($1 \le n \le 30$) indicating the number of cities, numbered 0 to $n-1$. After that are n lines each containing n integers specifying the distance in kilometers between cities: the j^{th} value on the i^{th} line indicates the distance when traveling directly from city i to city j. A value of -1 indicates there is no road directly connecting the two cities, and the distance from any city to itself is always 0; all other distances are positive and at most 1 000. There are at most 100 roads.

Following this is a line with a single integer r ($1 \le r \le 100$) indicating the number of previously executed routes. The next r lines each contain three integers s, d, and t, where s and d are the source and destination cities and t is how long the delivery from s to d took, in minutes.

Finally there is a line containing a single integer q ($1 \le q \le 100$) indicating the number of future delivery queries. The next q lines each contain two integers s and d giving the source and destination cities for the query.

You may assume that for each of the $r+q$ source/destination pairs in the input there is a unique minimum-distance route.

Output

Display a single line for each query containing the source and destination cities for that query, followed by the best low and high bounds on the estimate for the travel time, accurate to within an absolute or relative error of 10^{-6}.

Sample Input 1

```
3
0 50 -1
55 0 40
-1 40 0
1
0 2 120
3
0 1
1 2
1 0
```

Sample Output 1

```
0 1 50.0 80.0
1 2 40.0 70.0
1 0 55.0 110.0
```

Problem J
Spin Doctor
Time limit: 5 seconds

As an employee of the world's most respected political polling corporation, you must take complex, real-world issues and simplify them down to a few numbers. It isn't always easy. A big election is coming up and, at the request of Candidate X, you have just finished polling n people. You have gathered three pieces of information from each person, with the values for the i^{th} person recorded as:

- a_i – the number of digits of π they have memorized
- b_i – the number of hairs on their head
- c_i – whether they will vote for Candidate X

Unfortunately, you are beginning to wonder if these are really the most relevant questions to ask. In fact, you cannot see any correlation between a, b, and c in the data. Of course, you cannot just contradict your customer – that is a good way to lose your job!

Perhaps the answer is to find some weighting formula to make the results look meaningful. You will pick two real values S and T, and sort the poll results (a_i, b_i, c_i) by the measure $a_i \cdot S + b_i \cdot T$. The sort will look best if the results having c_i true are clustered as close to each other as possible. More precisely, if j and k are the indices of the first and last results with c_i true, you want to minimize the *cluster size* which is $k - j + 1$. Note that some choices of S and T will result in ties among the (a_i, b_i, c_i) triples. When this happens, you should assume the worst possible ordering occurs (that which maximizes the cluster size for this (S, T) pair).

Input

The input starts with a line containing n ($1 \le n \le 250\,000$), which is the number of people polled. This is followed by one line for each person polled. Each of those lines contains integers a_i ($0 \le a_i \le 2\,000\,000$), b_i ($0 \le b_i \le 2\,000\,000$), and c_i, where c_i is 1 if the person will vote for Candidate X and 0 otherwise. The input is guaranteed to contain at least one person who will vote for Candidate X.

Output

Display the smallest possible cluster size over all possible (S, T) pairs.

Sample Input 1

```
6
0 10 0
10 0 1
12 8 1
5 5 0
11 2 1
11 3 0
```

Sample Output 1

```
4
```

Sample Input 2

```
10
6 1 1
0 2 0
2 1 1
6 1 1
8 2 0
4 4 0
4 0 0
2 3 1
6 1 0
6 3 1
```

Sample Output 2

```
8
```

Sample Input 3

```
5
5 7 0
3 4 0
5 7 0
5 7 1
9 4 0
```

Sample Output 3

```
1
```

Problem K
String Theory
Time limit: 2 seconds

Nested quotations are great not only for writing literature with a complex narrative structure, but also in programming languages. While it may seem necessary to use different quotation marks at different nesting levels for clarity, there is an alternative. We can display various nesting levels using k-quotations, which are defined as follows.

A 1-quotation is a string that begins with a quote character, ends with another quote character and contains no quote characters in-between. These are just the usual (unnested) quotations. For example, `'this is a string'` is a 1-quotation.

For $k > 1$, a k-quotation is a string that begins with k quote characters, ends with another k quote characters and contains a nested string in-between. The nested string is a non-empty sequence of $(k-1)$-quotations, which may be preceded, separated, and/or succeeded by any number of non-quote characters. For example, `''All 'work' and no 'play'''` is a 2-quotation.

Given a description of a string, you must determine its maximum possible nesting level.

Input

The input consists of two lines. The first line contains an integer n ($1 \leq n \leq 100$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 100$), which describe a string as follows. The string starts with a_1 quote characters, which are followed by a positive number of non-quote characters, which are followed by a_2 quote characters, which are followed by a positive number of non-quote characters, and so on, until the string ends with a_n quote characters.

Output

Display the largest number k such that a string described by the input is a k-quotation. If there is no such k, display `no quotation` instead.

Sample Input 1	Sample Output 1
5 2 1 1 1 3	2

Sample Input 2	Sample Output 2
1 22	4

Sample Input 3	Sample Output 3
1 1	no quotation

This page is intentionally left blank.

Problem L
Swap Space
Time limit: 6 seconds

You administer a large cluster of computers with hard drives that use various file system types to store data. You recently decided to unify the file systems to the same type. That is quite a challenge since all the drives are currently in use, all of them are filled with important data to the limits of their capacities, and you cannot afford to lose any of the data. Moreover, reformatting a drive to use a new file system may significantly change the drive's capacity. To make the reformat possible, you will have to buy an extra hard drive. Obviously, you want to save money by minimizing the size of such extra storage.

You can reformat the drives in any order. Prior to reformatting a drive, you must move all data from that drive to one or more other drives, splitting the data if necessary. After a drive is reformatted, you can immediately start using it to store data from other drives. It is not necessary to put all the data on the same drives they originally started on – in fact, this might even be impossible if some of the drives have smaller capacity with the new file system. It is also allowed for some data to end up on the extra drive.

As an example, suppose you have four drives A, B, C, and D with drive capacities 6, 1, 3, and 3 GB. Under the new file system, the capacities become 6, 7, 5, and 5 GB, respectively. If you buy only 1 GB of extra space, you can move the data from drive B there and then reformat drive B. Now you have 7 GB free on drive B, so you can move the 6 GB from drive A there and reformat drive A. Finally, you move the six total gigabytes from drives C and D to drive A, and reformat C and D.

Input

The input begins with a line containing one integer n ($1 \le n \le 10^6$), which is the number of drives in your cluster. Following this are n lines, each describing a drive as two integers a and b, where a is the capacity with the old file system and b is the capacity with the new file system.

All capacities are given in gigabytes and satisfy $1 \le a, b \le 10^9$. (One thousand petabytes should be enough for everyone, right?)

Output

Display the total extra capacity in gigabytes you must buy to reformat the drives.

Sample Input 1

```
4
6 6
1 7
3 5
3 5
```

Sample Output 1

```
1
```

Sample Input 2	Sample Output 2

```
4
2 2
3 3
5 1
5 10
```
	`5`

Problem M
What Really Happened on Mars?
Time limit: 2 seconds

Real-time software in the Mars Pathfinder spacecraft suffered from an issue known as priority inversion. One technique to address this issue is to use the Priority Ceiling Protocol.

In this problem, you will simulate the execution of multiple tasks according to this protocol. The tasks share a collection of resources, each of which can be used by only one task at a time. To ensure this, resources must be locked before use and unlocked after use. Each task is defined by a start time, a unique *base priority*, and a sequence of instructions. Each task also has a *current priority*, which may change during execution. Instructions come in three types:

- *compute* – perform a computation for one microsecond
- *lock* k – lock resource k (which takes no processor time)
- *unlock* k – unlock resource k (which takes no processor time)

After locking a resource, a task is said to *own* the resource until the task unlocks it. A task will unlock only the owned resource it most recently locked, will not lock a resource it already owns, and will complete with no owned resources.

Each resource has a fixed *priority ceiling*, which is the highest base priority of any task that contains an instruction to lock that resource.

There is a single processor that executes the tasks. When the processor starts, it initializes its clock to zero and then runs an infinite loop with the following steps:

Step 1. Identify *running* tasks. A task is running if its start time is less than or equal to the current processor clock and not all of its instructions have been executed.

Step 2. Determine the current priorities of the running tasks and which of the running tasks are *blocked*. A running task T is blocked if the next instruction in T is to lock resource k and either resource k is already owned or at least one other task owns a resource ℓ whose priority ceiling is greater than or equal to the current priority of T. If T is blocked, it is said to be blocked by every task owning such k or ℓ. The current priority of a task T is the maximum of T's base priority and the current priorities of all tasks that T blocks.

Step 3. Execute the next instruction of the non-blocked running task (if any) with the highest current priority. If there was no such task or if a compute instruction was executed, increment the processor clock by one microsecond. If a lock or unlock instruction was executed, do not increment the clock.

The Priority Ceiling Protocol defined above has the following properties:

- Current priority is defined in terms of current priority and blocking, and blocking is defined in terms of current priority. While this may appear circular, there will always be a unique set of current priorities that satisfy the definitions.
- All tasks will eventually complete.
- There will never be a tie in step 3.

Input

The first line of the input contains two integers t ($1 \leq t \leq 20$), which is the number of tasks, and r ($1 \leq r \leq 20$), which is the number of resources. This is followed by t lines, where the i^{th} of these lines describes task i. The description of a task begins with three integers: the task's start time s ($1 \leq s \leq 10\,000$), its base priority b ($1 \leq b \leq t$), and an integer a ($1 \leq a \leq 100$). A task description is concluded by a sequence of a strings describing the instructions. Each string is a letter (`C` or `L` or `U`) followed by an integer. The string `C`n ($1 \leq n \leq 100$) indicates a sequence of n compute instructions. The strings `L`k and `U`k ($1 \leq k \leq r$) indicate instructions locking and unlocking resource k respectively.

No two tasks have the same base priority.

Output

For each task, display the time it completes execution, in the same order that the tasks are given in the input.

Sample Input 1

```
3 1
50 2 5 C1 L1 C1 U1 C1
1 1 5 C1 L1 C100 U1 C1
70 3 1 C1
```

Sample Output 1

```
106
107
71
```

Sample Input 2

```
3 3
5 3 5 C1 L1 C1 U1 C1
3 2 9 C1 L2 C1 L3 C1 U3 C1 U2 C1
1 1 9 C1 L3 C3 L2 C1 U2 C1 U3 C1
```

Sample Output 2

```
8
15
16
```