United Kingdom and Ireland Subregional Contest 2015

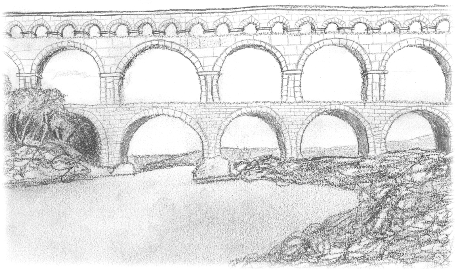
24th October 2015

Problems

Problems are not ordered by difficulty.
Do not open before the contest has started.

This page is intentionally left (almost) blank.

Problem A
Aqueduct Construction



After conquering Britannia, the great Roman general Agricola decided all of his new cities should take advantage of the natural springs found aplenty. He appointed his advisor Wessus Waterus to try to find a way to get each town a fresh supply of water.

There are many springs and many towns and between each are the natural hills and valleys of Roman Britain. Wessus doesn't want to waste the Imperial coin. He has been tasked with linking each town to a spring by a series of aqueducts using as little material as possible. Water, as we know, only flows downhill so any aqueduct must go from a higher point to a lower; intervening hills, springs and towns are no problem since they can be tunnelled through and on. The only requirement is that all aqueduct components start and end on hilltops.

Any spring must only serve one town, but the Romans are clever enough to find a way for aqueducts to pass each other. Roman engineering is excellent, but has its limits: aqueducts can only be of a limited length.

Input

- One line containing four integers: n, s, t and q where $0 < n \leq 500$ is the number of hills, $1 \leq s \leq 40$ is the number of springs, $1 \leq t \leq s$ is the number of towns and q ($1 \leq q \leq 3 \times 10^6$) is the maximum aqueduct length.
- N more lines, each giving the space-separated integers x_i, y_i, h_i: the coordinates and height of a hill ($0 \leq |x|, |y|, h \leq 10^6$). These hills are numbered 1 to n in the order given.
- One line containing s space-separated integers i_j ($1 \leq i_j \leq n$), each representing the number of a hill on which a spring can be found.
- One line containing t space-separated integers i_j ($1 \leq i_j \leq n$), each giving the number of a hill on which the town can be found.

Each hill may only have at most one spring or one town.

Output

Output one line with one real number, denoting the minimum total length of all aqueducts needed to supply each town with fresh water from its own unique spring or **IMPOSSIBLE** if there is no way to achieve this. Your answer should be correct up to an absolute or relative precision of 10^{-6}.

Sample Input 1	Sample Output 1
``6 2 2 8``` ```0 0 6``` ```3 4 7``` ```0 8 8``` ```6 8 8``` ```6 0 6``` ```6 4 8``` ```3 4``` ```1 5```	```20.396078```

Sample Input 2	Sample Output 2
``4 2 2 3``` ```1 3 2``` ```3 3 2``` ```2 1 1``` ```2 6 1``` ```1 2``` ```3 4```	```IMPOSSIBLE```

Problem B
Mountain Biking

Mount Snowdon, the tallest place in Wales, is a major attraction for mountain bikers far and wide. To take advantage of the popularity of this thrilling sport, an enterprising new venture plans to open several new bike repair shops throughout the rolling foothills.

The cunning small business owner's financial success relates directly to the velocity of the average biker: the faster a biker is going at the foot of the hill the more likely they are to encounter a problem and have to walk — or sometimes limp — into the shop.

Snowdon, like most mountains, has a very angular sort of shape. In fact, the profile of the mountain can be represented as N connected line segments pointing downward at various angles, each starting where the last left off. Given this highly scientific representation of the landscape, we need to find the likely speeds of bikers given that they may start off from the top of any of the N segments.

As we all know, a biker on a θ-degree slope from the vertical will accelerate at a rate of precisely $g \times \cos(\theta) \ \ ms^{-2}$ along the slope.

Input

- One line containing a positive integer N ($1 \leq N \leq 4$), the number of line segments making up the mountain, followed by a space and then a real number g ($1 \leq g \leq 100$), the coefficient of acceleration due to gravity.
- N more lines each containing two integers D_i and then θ_i ($1 \leq D \leq 10^4; 1 \leq \theta \leq 89$): the sloped distance in metres and absolute angle in degrees of this line segment from the vertical respectively. The segments are ordered from the top of the hill to its bottom.

Output

Each of the N lines of output should contain one real number: the velocity of a biker starting at the i^{th}-most segment from the top and finishing at the foot of the mountain.

Answers will be judged as correct if they are printed to within an absolute or relative difference of 10^{-6} from their exact values.

Sample Input 1

```
2 9
69 60
100 60
```

Sample Output 1

```
39.0
30.0
```

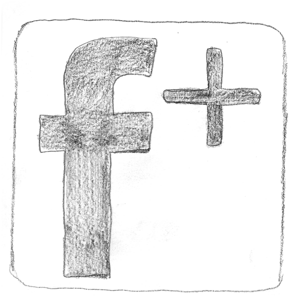
Sample Input 2

```
3 77
500 65
20 5
1000 80
```

Sample Output 2

```
249.70323
172.65601
163.52926
```

Problem C
Conversation Log



Popular social networking site My+Din is struggling to manage its many popular forums. Recent regulation requires the site to report users engaged in conversations about certain topics. The sheer number of users means manual monitoring is too costly and so the site has asked its many interns to come up with a solution.

One intern has theorized that conversations about any given topic will see the same key words used over and over. If the most used words can be identified perhaps manual investigation can be directed towards appropriate forums.

Input

- One line containing a single integer M ($1 \leq M \leq 10^4$), the number of messages.
- M more lines each beginning with a user's name of no more than 20 characters and continuing with the content of that user's message all in lower case. The total number of characters across all messages, including spaces, will not exceed 2×10^6.

Output

Several words, one per line, listing the words used by every single user on the forum, ordered from most to least used and in case of a tie in alphabetical order. If there are no such words, output **ALL CLEAR**.

Sample Input 1

```
8
Jepson no no no no nobody never
Ashley why ever not
Marcus no not never nobody
Bazza no never know nobody
Hatty why no nobody
Hatty nobody never know why nobody
Jepson never no nobody
Ashley never never nobody no
```

Sample Output 1

```
no
nobody
never
```

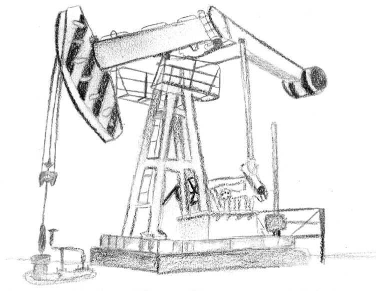
Sample Input 2

```
2
Villain avast
Scoundrel ahoy
```

Sample Output 2

```
ALL CLEAR
```

Problem D
Slant Drilling



AcmeCorp is always looking to expand its drilling operations.

Their latest target is a largely uninhabited area in the North of the country, known as Crossland. Since the pocket of oil is quite deep already, and new drill bits are expensive, Crossland's oil company would like to minimise the distance drilled — which may not be vertical — to get to the valuable spot.

The elevation-contoured survey map we have obtained marks the oil pocket at (0,0), 0 metres above sea level.

Input

- One line containing one positive integer N, the number of polygon-shaped contour lines.
- N more lines each containing two integers, H_0 and H_1 ($0 \leq H_0, H_1 \leq 10^6$), the height of land in metres above sea level outside and inside the contour respectively, followed by a positive integer M_i representing the number of vertices of this contour.
- The remaining $2 \cdot M_i$ integers on each line are distinct coordinate pairs (x_j, y_j) ($-10^6 \leq x, y \leq 10^6$) denoting the j^{th} point on the contour. No two contour lines touch, nor does a contour line touch the point (0,0).

The total number of points across all contours will not exceed 10^5, and it is guaranteed that the outer height of a contour is always equal to the inner height of its containing contour, if one exists.

Output

The first and only line of output should contain one real number indicating the closest slanted distance from the surface to the target. Your answer should be correct up to an absolute or relative precision of 10^{-6}.

Sample Input 1

```
2
5 8 4 -2 0 0 -3 2 0 0 3
8 10 4 -1 -1 1 -1 1 1 -1 1
```

Sample Output 1

```
5.2696518641
```

Sample Input 2

```
1
8 9 3 -14 -14 14 -14 0 20
```

Sample Output 2

```
9
```

Sample Input 3

```
1
2 1 8 -4 -4 -1 -3 -2 2 2 2 1 -3 4 -4 3 4 -3 4
```

Sample Output 3

```
1.8605210188
```

Problem E
Rainfall

About to leave the university to go home, you notice dark clouds packed in the distance. Since you're travelling by bicycle, you're not looking forward to getting wet in the rain. Maybe if you race home quickly you might avert the rain. But then you'd get wet from sweat...

Facing this dilemma, you decide to consider this problem properly with all data available. First you look up the rain radar image that shows you precisely the predicted intensity of rainfall in the upcoming hours. You know at what time you want to be home at the latest. Also, you came up with a good estimate of how much you sweat depending on your cycling speed. Now the question remains: what is the best strategy to get home as dry as possible?

The rain is given for each minute interval in millilitres, indicating how wet you get from cycling through this — note that you can cycle just a fraction of a whole minute interval at the start and end of your trip: then only that fraction of the rain during that interval affects you. Sweating makes you wet at a rate of $s = c \cdot v^2$ per minute where v is your speed in km/h and c is a positive constant you have determined. You have to cover the distance to your home in a given time (you don't want to wait forever for it to become dry), but otherwise you can choose your strategy of when to leave and how fast to cycle (and even change speeds) as you wish. What is the least wet you can get from the combination of rain and sweat?

Input

- One line containing a single positive integer T ($0 < T \leq 10\,000$), the number of minutes from now you want to be home by at the latest.
- Another line with two positive floating point numbers: c ($0.01 \leq c \leq 10$), the constant determining your sweating and d ($1 \leq d \leq 50$) the distance from university to home in kilometres.
- T more lines, where each line contains an integer r_i ($0 \leq r_i \leq 100$) the number of millilitres of rain during the i-th minute interval (zero-based).

Output

On a single line print a floating point number: the number of millilitres of rain and sweat you get wet from when optimally planning your cycle home. Your answer should be correct up to an absolute or relative precision of 10^{-6}.

Sample Input 1	Sample Output 1
5 0.1 2.0 0 0 0 0 0	288

Sample Input 2	Sample Output 2
30 0.01 2 0 0 0 100 100 100 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	24

Problem F
Physiognomy

A properly designed room is, as we all know, well-lit.

In keeping with the teachings of *Feng Shui*, you have placed a number of lamps around a newly-designed room at strategic places to give it a friendlier air.

Some of the lights project *positive* energy, and the rest give out only *negative* energy. Luckily, your neighbourhood guru knows all about this, and will gladly help you to keep the delicate energy balance... For a small fee.

The *balancing line* of a particular lighting arrangement is the shortest continuous closed circuit dividing energy sources into two parts, those inside the line and those outside the line, such that the sum of energies inside and outside is equal — cancelling out any negative effects.

What is the length of this line?

Input

- A line with one positive integer, N ($2 \leq N \leq 12$), the number of lamps.
- N more lines, each containing three space-separated integers x_i and y_i ($1 \leq x_i, y_i \leq 99$) giving the coordinates of the i^{th} lamp expressed in centimetres from the corner of the room, and e_i ($-2000 \leq e_i \leq +2000$), the energy contribution of this lamp. A lamp placed at (x, y) has a square footprint that fits inside the square with opposite corners $(x-1, y-1)$ and $(x+1, y+1)$ with a tiny amount of space to spare.

It is guaranteed that no two lamps will have overlapping footprints.

Output

Write one real number: the length of the shortest continuous line dividing the positive and negative energy sources in the room. Your answer should be correct up to an absolute or relative precision of 10^{-6}.

If no such line exists, output **IMPOSSIBLE** instead.

Sample Input 1	Sample Output 1
4 10 10 5 10 20 5 20 10 5 20 20 5	28

Sample Input 2	Sample Output 2
4 10 10 5 10 20 1 20 10 12 20 20 8	36.2842712475

Sample Input 3	Sample Output 3
6 1 1 15 5 1 100 9 1 56 1 5 1 5 5 33 9 5 3	28.970562748

Sample Input 4	Sample Output 4
8 4 4 1 4 6 1 4 8 1 6 6 14 8 4 1 8 6 1 8 8 1 99 6 -8	32

Sample Input 5	Sample Output 5
2 4 4 2 8 8 3	IMPOSSIBLE

Problem G
Drink Responsibly

The University of Lagado is organising events for the upcoming Fresher's week and has been told — much to the surprise of its staff — that some of the undergraduates may enjoy a beer tasting. While sourcing a wide variety of drinks for the students to taste, the university realised that in the interests of safety there should be a limit on the alcohol consumption of any student, enforced by a strict limit on the amount any individual is allowed to spend.

In common with many popular establishments, the drinks with varying strengths are served in varying amounts: Either a litre, a half litre or a third of a litre to limit possible intoxication.

The students are looking forward to the event, but in order to make the most of their money and still be bright-eyed and bushy tailed for the first week of morning lectures, they don't wish to get too drunk. How can the students spend all their money and consume in full their self-imposed alcohol limit for the night?

Input

- One line containing three numbers:
  - m ($0.00 \leq m \leq 10.00$), the amount of money they can spend to two decimal places;
  - u ($0.0 \leq u \leq 20.0$), the number of units they aim to drink to one decimal place;
  - d ($1 \leq d \leq 8$), the number of different drinks available.
- Another d lines, each containing:
  - up to 20 lowercase latin letters (the name of the drink);
  - an integer between 0 and 100 (its strength as a percentage);
  - its size (either '1/1' for a litre, '1/2' for a half or '1/3' for a third);
  - a real number to two decimal places; its cost.

Name, strength, price and cost will be separated by spaces.

Output

If the students' aims are possible, write one or more lines, each listing the name of the drink purchased followed by the positive integer count of how many drinks of that type were bought. If there are several solutions, output any.

Otherwise, output a single line containing the word **IMPOSSIBLE**.

Sample Input 1

```
10.00 9.0 2
fire 2 1/1 4.00
water 10 1/2 2.00
```

Sample Output 1

```
fire 2
water 1
```

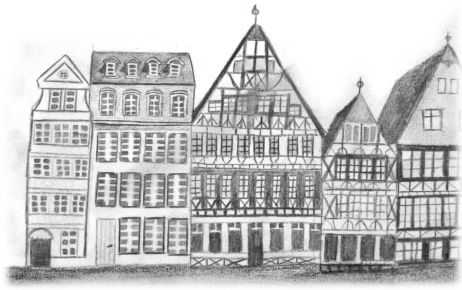
Sample Input 2

```
2.00 3.0 3
firewater 1 1/1 1.00
windwater 1 1/1 1.00
earthwater 1 1/1 1.00
```

Sample Output 2

```
IMPOSSIBLE
```

Problem H
Sunlight



A core right in Roman tenancy law was the availability of sunlight to everybody, regardless of status. Good sun exposure has a number of health benefits, many of which were known even in those ancient times.

The first act of a Roman city plan reviewer, then, is to survey the proposed structures to measure how well they distribute this precious resource. Given any one avenue of buildings arranged West-to-East, the number of hours for which each building is exposed to sunlight needs to be determined.

For the purpose of simplicity, the number of hours a building is in sunlight is proportional to the fraction of the 180 degrees of sky visible from its top. Thanks in no small part to the marvels of ancient engineering (and also to the strict nutritional regimes of old) you may assume each building is infinitesimally thin.

Input

- One line containing one integer N ($1 \leq N \leq 2 \times 10^5$): the number of buildings.
- N further lines each containing two space-separated integers X_i and H_i ($1 \leq X, H \leq 10^9$), the location and height respectively, in metres, of the i^{th}-most building from the west.

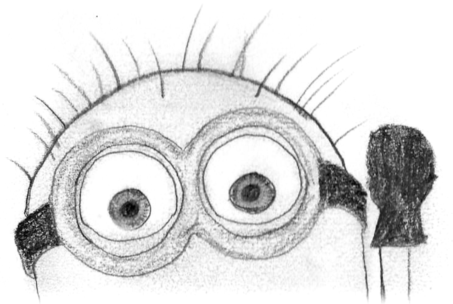
Output

On each of the N lines of output, write one real number to at least 4 decimal places of accuracy: the number of hours for which the peak of the i-th building is bathed in sunlight.

Sample Input 1	Sample Output 1
<pre>4 1 1 2 2 3 2 4 1</pre>	<pre>9.0000 12 12.00000 9.0</pre>

Sample Input 2	Sample Output 2
<pre>5 100 50 125 75 150 100 175 125 200 25</pre>	<pre>9.0 9.0 9.0 12.0 6.9357496</pre>

Problem I
Nimionese



Nimions speak a funny form of language.

Whichever word they are trying to say, from which ever language, it doesn't quite come out the same. There are several rules for converting words, from any language, to nimionese.

For any word:

- **All nimion words start with 'hard' consonants** — [b, c, d, g, k, n, p, t], so you must replace each first letter with the nearest one (choose the option nearest to 'A' if there is a tie).
  - "Each" becomes "Dach".
- **Any hard consonant** in subsequent syllables after the first one is remarkably better if it is replaced with the same consonant as the one at the start of the word.
  - "Hip-po" becomes "Gip-go".
- **No word ends in a hard consonant.** You must add an 'ah', 'oh' or 'uh' at the end, whichever is nearest, rounding toward 'A' in the case of a tie, to the last hard consonant in the word.
  - "Dog" becomes "Dogah"
  - "Hip" becomes "Gipoh".

Input

The only line of input contains a sentence of between 1 and 50 words and up to 10^4 symbols, including single whitespace characters (' ') between words and the dashes ('-') between each syllable.

Apart from dashes, the sentence will contain solely lower-and-upper-case Latin letters — and only the first letters of words can be upper-case.

Output

Write to standard output the same sentence from the input, translated to nimionese. Remove any dashes before printing.

It is guaranteed that the output will fit within $5 \cdot 10^4$ characters.

Sample Input 1

```
I love ba-na-na
```

Sample Output 1

```
Gah kove bababa
```

Sample Input 2

```
Cat-a-pil-lar
```

Sample Output 2

```
Catacillar
```

Sample Input 3

```
Co-ding is the best
```

Sample Output 3

```
Cociccah gs the bestuh
```

Sample Input 4

```
I rock at co-lour-ing
```

Sample Output 4

```
Gah pockoh btuh colouriccah
```

Problem J
Jelly Raid

The full-board residents of the famous but misnamed Adephagia School For Boys feel hard done by. Recent savings have seen dinnertime puddings cut back to a single biscuit while the masters finish their meals with fine raspberry jelly every night. Although the choice of custard cream or bourbon is not a bad one, the children hunger for the raspberry-filled delight and have planned a midnight raid.

A child must make it from their dorm, across the school to the kitchen without being seen by the patrolling masters. In any turn a child can stand still, or move to a horizontally or vertically neighbouring cell. Each turn the patrolling masters will move a step along their route. When they reach the end of their route, they will turn and continue to retrace their steps indefinitely. A child will be caught if the child and a master are on the same row or column with no blocked area between them.

If the child can make it to the fridge in the kitchen unseen, we recognise they can either make it back to their bed unseen or get caught fully of jelly and not care. See the first example, where the child can reach the fridge after 26 turns, even though it gets spotted at that same time.

Input

- One line consisting of two positive integers r c ($1 < r, c \leq 60$), the size of the school in rows and columns.
- One line consisting of two pairs of bracketed positive integers, the row and column coordinates of the starting bed in the dormitory and the fridge in the kitchen respectively.
- Another r lines, each containing a string of c characters with the following meaning:
  - '`.`': Walkable area
  - '`#`': Blocked area
- One line consisting of a single integer p ($1 \leq p \leq 200$) representing the number of patrolling masters.
- Another p lines, each a size-prefixed list of between 1 and 7 space-separated bracketed (r, c) coordinates representing a contiguous path for the master to patrol.

All 'special' coordinates (locations of the bed, fridge, and masters) will be marked as walkable on the map.

Output

On a single line print the minimum number of turns for the child to reach the jelly. If the child cannot reach the jelly, output **IMPOSSIBLE**.

Sample Input 1

```
5 5
(2 5) (5 3)
.....
.#.#.
.#.#.
....#
.#.##
1
6 (4 2) (4 3) (3 3) (2 3) (1 3) (1 2)
```

Sample Output 1

```
26
```

Sample Input 2

```
5 4
(1 4) (5 4)
....
..#.
###.
....
###.
2
2 (2 2) (2 1)
4 (4 1) (4 2) (4 3) (4 3)
```

Sample Output 2

```
IMPOSSIBLE
```

Problem K
Call a Cab

Tourists in the numerous Welsh valleys are in need of an IT solution to their transportation troubles. They want to see all the local Points of interest (POIs) in a specific order already set out by their trusty tour guides.

Visitors have to rely on various types of transportation: car, rickshaw, donkey cart, etc. These are only available on call, which is unfortunate as the poor mobile reception in the depths of the idyllic valleys means transportation type can only change at a POI.

Further, a driver will only offer tourists transportation between points p_i, p_{i+1}, $\ldots$, p_{j-1}, p_j under the following conditions:

- **Minimum distance:** If the distance is less than a given threshold d_{min}, the itinerary wouldn't be worth the time of the driver. That is, the total distance of the itinerary: $d_i + d_{i+1} + \ldots + d_{j-1}$, with d_m the distance between p_m and p_{m+1}, has to be at least d_{min}.
- **Maximum heading range:** Not going straight is perceived as annoying by the cabbies, so the directions traveled can only vary within at most a certain integer amount of r_{max} degrees.

What the tourists want is a transportation switching scheme, which is a list of increasing indices $s_0 \ldots s_k$ where points p_{s_i} are the locations to switch the type of transportation (the same transportation type can be used more than once, but another instance of the same type will need to be hailed once the original driver has had enough).

Input

- One line containing the number of modes of transportation t ($1 \leq t \leq 200$) followed by the number n ($1 \leq n \leq 5 \times 10^4$) of points we visit.
- The next t lines each describe the i^{th} transportation type with two non-negative integers. The first integer d_{min} ($0 \leq d_{min} \leq 10^6$) is the minimal distance of each itinerary of this type. The second integer a ($0 \leq a \leq 3.6 \times 10^5$) is the maximal heading range in thousandths of a degree.
- The next $n - 1$ lines each contain two integers d_i and h_i ($0 \leq d_i \leq 10^6$; $-1.8 \times 10^5 < h_i < +1.8 \times 10^5$), relative distance and angle from the $i - 1^{th}$ point in thousandths of a degree respectively.

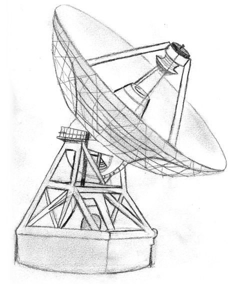
Output

Write one line containing one number k: the minimal number of times we have to call for a new type of transportation to visit all n points in the given order, if this is possible. If not, output **IMPOSSIBLE**.

Sample Input 1	Sample Output 1
4 4 100 30000 200 20000 300 10000 400 0 50 10000 75 20000 400 -40000	2

Sample Input 2	Sample Output 2
1 3 20 50000 100 10000 10 -60000	IMPOSSIBLE

Problem L
Telescope



Photography is a deceptively simple subject to the naïve outsider.

In truth, viewing angles, saturation levels, contrast, proper lighting and focusing all play a part. Today, we are most concerned with the problem of focus.

The proudest space telescope of all, Inquisition IV, launched into space over a decade ago and is starting to show its age in the pictures sent back. Not only is the resolution low by modern standards, all images are painfully blurry! Each pixel in the camera viewport is reported as the average of all those around it in an $N \times N$ box, rounded downwards into an integer.

To secure more funding for future research (and, if budget allows it, some lens wipes for the next telescope) you need to prove you know what's out there, specifically the number of stellar bodies visible to the telescope. As is well-known by now, stellar bodies are very bright: in fact it is guaranteed that each registers as a region of horizontally and vertically connected pixels of total brightness to the sensor, while everything else registers as fully black.

Handily the telescope has framed the entire "scene", including capturing all of the surrounding blurriness on camera. This means that any pixels outside the capture region can also be considered as totally black.

Input

- One line containing three integers N, R and C ($1 \leq N \leq 99 \leq R, C \leq 1000; N \equiv 1 \mod 2$) indicating the blur box width and rows and columns registered by the camera respectively.
- Another R lines, each containing C space-separated hexadecimal integers $L_{r,c}$ (0x0000 $\leq L_{r,c} \leq$ 0xFFFF) where a value of 0x0000 is total blackness and a value of 0xFFFF is total whiteness.

Output

A single integer: the number of stellar bodies visible in the snapshot.

Sample Input 1

```
1 5 6
0000 FFFF 0000 0000 0000 0000
FFFF FFFF 0000 FFFF FFFF 0000
0000 0000 0000 FFFF 0000 0000
0000 FFFF FFFF FFFF FFFF 0000
0000 0000 0000 0000 0000 0000
```

Sample Output 1

```
2
```

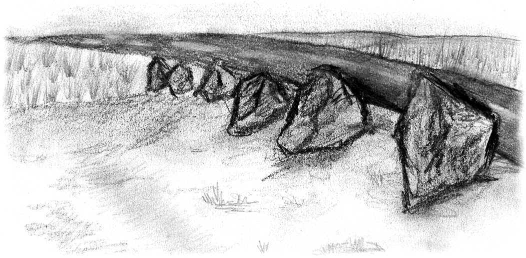
Sample Input 2

```
3 5 6
1C71 1C71 1C71 0000 0000 0000
1C71 1C71 1C71 0000 0000 0000
1C71 1C71 1C71 1C71 1C71 1C71
0000 0000 0000 1C71 1C71 1C71
0000 0000 0000 1C71 1C71 1C71
```

Sample Output 2

```
2
```

Problem M
Milestone Counter



Driving through the Irish countryside, one frequently sees enigmatic small grey stones sitting by the wayside, spaced about a kilometre and a half apart. As it turns out, these stones once had a purpose: they were **milestones**, intended to demarcate this venerable unit of measurement.

Being so old and, crucially, collectible by magpies and larger scamps alike, not all of these stones have remained.

Passing by some more of these tattered markers at a constant but unknown speed, you may still be able to gain some information from their placements. For example, since you started counting you have passed exactly M remaining stones; how fast could you have been driving?

Input

- One line containing two positive integers, M and N ($2 \leq M \leq N \leq 10^3$): the number of consecutive stones you noticed and the total number of stones along the road respectively.
- One line containing M distinct non-negative integers $T_{1..M}$ in ascending order — the times at which you passed stones in hours ($0 \leq T_i \leq 10^{15}$).
- One line containing N distinct non-negative integers $X_{1..N}$ in ascending order — the distances along the road of each milestone ($0 \leq X_i \leq 10^{15}$) in miles.

Output

Output two lines:

- First, the number of distinct possible speeds at which the car could have been travelling.
- Second, a space-separated list of all of the possible distances between the first milestone you saw and the second milestone you saw, in increasing order.

Sample Input 1

```
4 12
1 2 4 5
6 8 12 18 26 28 30 34 36 37 39 40
```

Sample Output 1

```
2
1 2
```

Sample Input 2

```
5 10
1 2 3 4 5
0 1 2 3 4 5 6 7 8 9
```

Sample Output 2

```
1
1
```

Sample Input 3

```
3 6
1 2 4
11 12 15 19 24 30
```

Sample Output 3

```
0
```

Sample Input 4

```
2 3
1123456789000 1123456789007
5 8 13
```

Sample Output 4

```
2
3 5
```