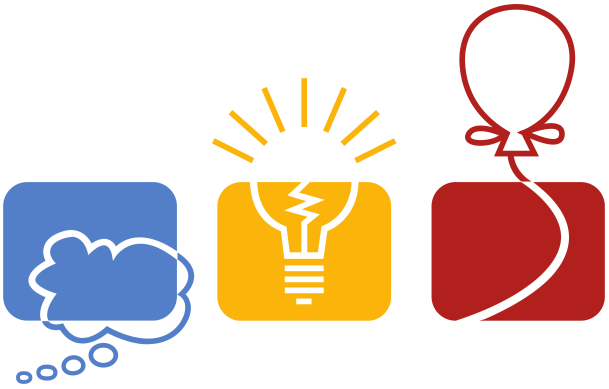
United Kingdom and Ireland Programming Contest 2024



Problems

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

This page is intentionally left (almost) blank.

Problem A
Amalgram

An **anagram** is any arrangement of the letters of a word in which each letter of the alphabet occurs exactly as many times as in the original. For example, `clarinets` is an anagram of `larcenist`.

An **amalgram** is any arrangement of the letters of **two** words in which each letter of the alphabet occurs **at least** as many times as in either of the originals. For example, `administration` is an amalgram of `mantis` and `raisin`, although not the shortest possible because the letter `d` appears in neither.

Given two words, invent an amalgram for them that contains as few letters as possible.

Input

- One line containing lowercase Latin letters representing the word a ($1 \leq |a| \leq 10^6$).
- One line containing lowercase Latin letters representing the word b ($1 \leq |b| \leq 10^6$).

Output

Output a minimally-long sequence of letters that represents an amalgram of a and b. If there are multiple answers, you may output any of them. Your answer will be judged as correct if it contains at least all of the letters of a and all of the letters of b, and there is no other possible answer that could be shorter.

Sample Input 1	Sample Output 1
`hello` `world`	`wordhell`

Sample Input 2	Sample Output 2
`unclear` `instructions`	`lensrustication`

Sample Input 3	Sample Output 3
`boring` `boring`	`boring`

This page is intentionally left (almost) blank.

Problem B
Budget Analysis

You are an analyst, studying the relationship between advertisement budget spending (denoted by x) and sales (denoted by y) over the period of n months. More specifically, for every month of time from 1 to n you have the value of spending x_i and sales y_i.

To quantify the relationship you are using linear regression with regularisation, which means that you are modelling y as $y = Kx + B$, where K and B are real numbers minimising the penalty function:

$$p(K, B) = \sum \left((K \cdot x_i + B - y_i)^2 \right) + \lambda \cdot (K^2 + B^2)$$

(Note: this is the standard penalty function for L2 regularised linear regression.)

For the report requested by your manager, you need to make several predictions. More specifically, you have a list of prediction queries, each described by four numbers — L_j, R_j, λ_j and X_j. To process such a query you need to perform the following steps:

- take the spending and sales values for the months from L_j to R_j inclusive;
- find the coefficients K and B, which minimise the penalty function for the given regularisation coefficient λ_j;
- plug the X_j into the resulting model and compute the prediction.

You are given the ads spending and sales data, and the prediction queries descriptions. You are to process the queries and output the predictions.

Input

First line of the input file contains an integer number n ($2 \le n \le 10^6$) denoting the number of months in the period you are studying.

Each of the following n lines describes one month and contains two non-negative real numbers x_i and y_i not exceeding 10. They denote the budget spending and sales in the corresponding month.

The following line contains an integer number m ($1 \le m \le 10^6$) denoting the number of predictions to be made. Each of the following m lines contains four numbers: L_j, R_j, $lambda_j$ and X_j ($1 \le L_j < R_j \le n$, $0 \le \lambda_j, X_j \le 10$). First two of them are integers, the remaining are real.

Output

For each prediction query output one real number on a separate line — the predicted sales assuming the advertisement spending is X_j and the linear model has been fitted on months from L_j to R_j using L2-regularisation with λ_j regularisation coefficient. The output must be accurate to an absolute or relative error of at most 10^{-6}.

Sample Input 1

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

Sample Output 1

```
11
4.90566037735849125
```

Sample Input 2

```
3
1 1.0
2 2.1
3 2.8
3
1 2 0 1.5
2 3 0 2.5
1 3 0 1.5
```

Sample Output 2

```
1.55
2.45
1.516666666666667
```

Problem C
Cross Country

Cross-country running is a sport in which contestants run a race on an open-air course over natural terrain. To record contestants' progress, the organisers set up RFID checkpoints that each span a line across part of the course.

A contestant has finished the race once they go through all of the checkpoints in order from 1 to n. Crossing a checkpoint out of order conveys no advantage or penalty to a runner, as they simply have to cross it again later at the right time. Thus, for example, a runner may choose to cross a checkpoint once and then immediately cross it again in another direction if it leads to a quicker finish.

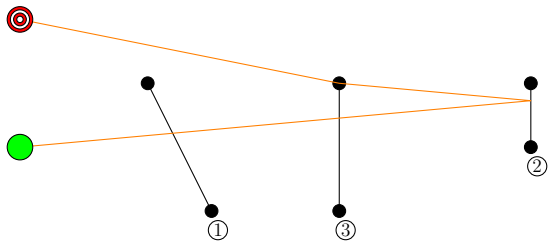


Figure C.1: Optimal running route for the course given in sample input 3.

Your objective is to find the shortest distance one has to run to finish the race, so that we can use this as the official distance of the course.

Input

- One line containing the number of checkpoints, n ($1 \leq n \leq 16$).
- One line containing the start coordinate of the race, x_s and y_s ($-10^6 \leq x, y \leq 10^6$).
- n further lines, the ith of which contains the two integer coordinate of the ith checkpoint's endpoints, $x_{ai} y_a x_b y_b$ ($-10^6 \leq x, y \leq 10^6$).
- One line containing the end coordinate of the race, x_t and y_t ($-10^6 \leq x, y \leq 10^6$).

All of the checkpoints have non-zero length; however, they may overlap either with each other or with the start and finish points.

Output

Output the shortest distance you can run to go visit all of the checkpoints in the right order, regardless of whether you touch some of the checkpoints multiple times or in the wrong order along the way.

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

Sample Input 1

```
2
0 1
10 0 10 2
20 2 20 0
30 1
```

Sample Output 1

```
30
```

Sample Input 2

```
4
5 5
10 1 8 -1
12 3 13 0
18 3 17 0
20 1 22 -1
25 5
```

Sample Output 2

```
22.80624847
```

Sample Input 3

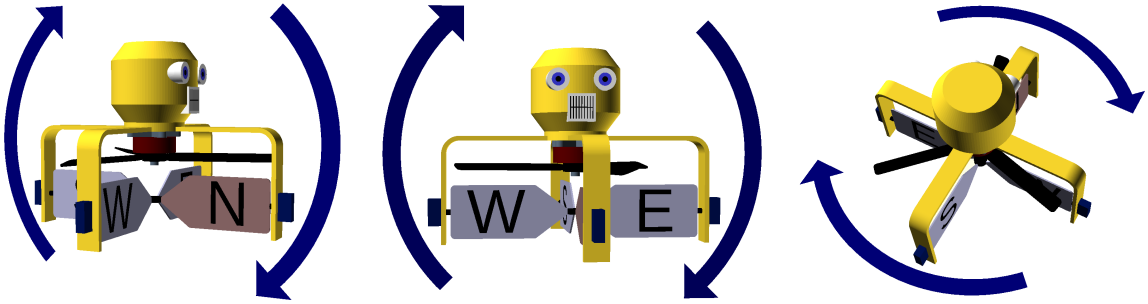
```
3
0 0
3 -1 2 1
8 0 8 1
5 -1 5 1
0 2
```

Sample Output 3

```
16.144380531
```

Problem D
Drone Control

You are designing a controller for an interesting aircraft called the *Single Copter*. It only has one propeller, but the outgoing air flow is further shaped by four *flaps* that control three Euler angles (pitch, roll and yaw) that help maintain the requested orientation of the craft. Each of these flaps can assume any angle requested by the flight controller, and the effects of the flaps being at certain angles should translate to exerting the requested forces on pitch, roll and yaw.



Pitch Roll Yaw

Figure D.1: Pitch, roll and yaw on a *Single Copter*

Define the angles of the flaps to be n, e, s, and w (for "north", "east", "south" and "west" respectively). The forces in the directions of pitch, roll and yaw are defined by the following equations:

$$p = e - w$$
$$r = n - s$$
$$y = n + e + s + w$$

As there are four variables and three constraints, you decided that, from the perspective of aerodynamics, it makes sense to make the maximum of the flap angles as small as possible, that is, you additionally want to minimise $\max\{|n|, |e|, |s|, |w|\}$.

Find the best parameters to send to the *Single Copter* to achieve the desired pitch, yaw, and roll.

Input

- One line containing the number q, $1 \leq q \leq 10^4$, the number of requests to follow.
- i further lines, each containing three real numbers p_i, r_i, y_i ($-1 \leq p_i, r_i, y_i \leq +1$).

Output

Output q lines. In the i-th line, output the solution for the i-th request, four numbers n_i, e_i, s_i, w_i, separated by whitespace.

Your answer will be considered correct if the resulting pitch, roll and yaw differ by at most 10^{-6} from the requested ones, and the maximum of the absolute values of flap outputs does not exceed the true value by more than 10^{-6}.

Sample Input 1	**Sample Output 1**
8 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 1 1	0.0 0.0 0.0 0.0 0.0 0.5 0.0 -0.5 0.5 0.0 -0.5 0.0 0.25 0.25 0.25 0.25 0.5 0.5 -0.5 -0.5 0.5 0.5 0.5 -0.5 0.5 0.5 -0.5 0.5 0.75 0.75 -0.25 -0.25

Problem E
Eradication Sort

The members of the No-Weather-too-Extreme Recreational Climbing society completed their first successful summit seven years ago to this day!

At the time, we took a picture of all the members standing together in one row. However, the photograph looks messy, as the climbers were not standing in order of height, and we have no way to reorder them.

We will need to cut some of the climbers out of the picture.

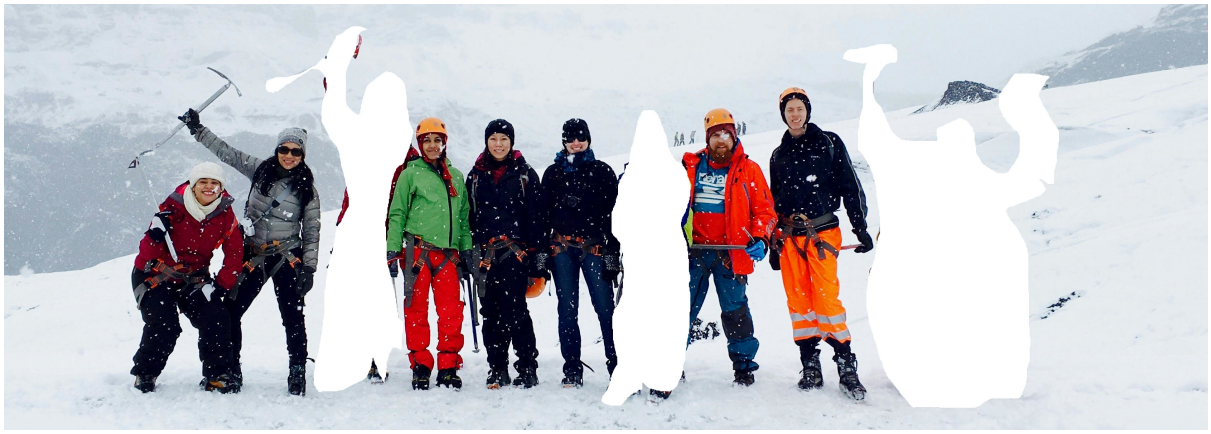


Figure E.1: This picture of 7 (formerly 11) climbers was edited to solve Sample Input 3.

An optimal solution minimises the size and number of visible gaps in the photo. We define the *cost* as the sum of the squares of the lengths of gaps left in the edited photo. For example, if two individual climbers are removed from the photo and one pair of adjacent climbers are removed, the total *cost* is $1^2 + 1^2 + 2^2 = 6$.

Find the minimum possible cost you can reach by removing climbers.

Input

- The number of people in the photo n ($1 \leq n \leq 10^6$).
- n integers representing the heights of people in the photo, $h_1 \ldots h_n$ ($0 \leq h \leq 10^6$).

Output

Output the minimum *cost* achieved by removing climbers from the photo, such that the remaining climbers in the photo make a **non-decreasing** sequence.

Sample Input 1	Sample Output 1
7 1 2 3 0 5 6 7	1

Sample Input 2

```
9
4 5 6 4 2 3 6 6 6
```

Sample Output 2

```
8
```

Sample Input 3

```
11
3 6 12 7 7 7 6 8 10 5 5
```

Sample Output 3

```
6
```

Problem F
Finding Suspicious Proteins

Little Claire studies proteins, which are sequences of amino acids. There are 20 amino acids from which proteins are built. While amino acids all have proper names, such as *alanine* or *glycine*, they are often denoted by single letters, so that proteins can be seen as sequences of different lengths, such as `DTASDAAAAAALTAABAAAAAKLTABBAAAAAAATAA`, `TIFLQQQQQQQQQQ` or even maybe `RICKRQLL`.

Comparing two proteins can be difficult, because they may contain active sites, which determine their function in a cell, and less important parts of the sequence. Recent advances in artificial neural networks made it possible to train a network that, given a protein, outputs a sequence of l numbers, where each number roughly corresponds to a feature of a protein that correlates with its possible functions in a cell. Such a sequence is called an *embedding*.

Claire is particularly interested in *suspicious* proteins, those which are really different from others. For this purpose, she considers the so-called *Manhattan distance* between embeddings of proteins. For two protein embeddings p and q of length l, the distance $\mathcal{D}(p, q)$ is computed as follows:

$$\mathcal{D}(p, q) = \sum_{i=1}^{l} |p_i - q_i|,$$

where p_i is the i-th element of the embedding p.

Claire wants to find k suspicious proteins in the given list of n proteins. As a baseline for her studies, Claire wants to use the following greedy algorithm:

- Find a protein $p^{(1)}$ which is the most distant from the first protein in the list.
- The second protein, $p^{(2)}$, is chosen as the most distant protein from $p^{(1)}$.
- The third one, $p^{(3)}$, is chosen so that $\min\{\mathcal{D}(p^{(1)}, p^{(3)}), \mathcal{D}(p^{(2)}, p^{(3)})\}$ is maximum possible. That is, it must be far away from *both* previously chosen proteins.
- All subsequent proteins $p^{(i)}$, $4 \leq i \leq k$, are chosen in a similar way: the minimum of the distances to all the previously chosen proteins should be maximum possible.

Note that, in the case of ties, the first matching protein in the list must be chosen.

Claire's implementation works nicely for small numbers n and k, but becomes too slow as they increase. You must find a way to optimise this.

Input

The first line contains three numbers n ($3 \leq n \leq 10^4$), l ($1 \leq l \leq 100$) and k ($2 \leq k \leq \min\{n, 256\}$): the overall number of proteins, the length of each protein embedding, and the number of proteins to choose.

Each of the following n lines starts with a protein identifier, which is a sequence of at least one and most ten capital letters and/or numbers. Then, separated by whitespace,

come l single-digit integer numbers $v_{1...l}$ ($0 \leq v \leq 9$), which define the embedding of the protein. All protein identifiers will be different.

Output

Output the identifiers of k chosen proteins, one per line, in their respective order ($p^{(1)}$ to $p^{(k)}$).

Sample Input 1

```
4 2 2
FIRST 3 4
SECOND 1 2
THIRD 8 7
FOURTH 5 6
```

Sample Output 1

```
THIRD
SECOND
```

Sample Input 2

```
6 5 3
1OGLOBIN 1 1 1 1 1
GLU10 9 9 9 9 9
8EIN 8 9 8 9 9
COLLA6EN 6 5 4 3 2
7ILK 3 4 5 6 7
0LBUMIN 1 2 0 2 1
```

Sample Output 2

```
GLU10
1OGLOBIN
7ILK
```

Problem G
Word Search

Find parts of a 2d grid matching a 2d word.

Input

- One line containing the number of rows and columns in the search key, r_k and c_k ($1 \leq r, c \leq 2000$).
- r_k further lines, each containing c_k Latin characters comprising a row of the search key.
- One line containing the number of rows and columns in the haystack, r_h and c_h ($r_k \leq r_h \leq 2000$, $c_k \leq c_h \leq 2000$).
- r_h further lines, each containing c_h Latin characters comprising a row of the search key.

Output

Illustrate the matching areas of the haystack by printing a grid of the same size. In locations that are part of at least one match, print the original character from the haystack. In other cases, print a full-stop "." character.

Sample Input 1

```
3 3
ghi
lmn
qrs
5 5
abcde
fghij
klmno
pqrst
uvwxy
```

Sample Output 1

```
.....
.ghi.
.lmn.
.qrs.
.....
```

Sample Input 2

```
1 2
ab
6 4
abba
baab
abba
baab
abba
baab
```

Sample Output 2

```
ab..
..ab
ab..
..ab
ab..
..ab
```

Sample Input 3

```
4 1
n
a
n
a
7 6
ananan
nanana
ananan
nanana
ananan
nanana
batman
```

Sample Output 3

```
.n.n.n
nanana
ananan
nanana
ananan
.a.ana
....a.
```

Sample Input 4

```
2 2
oo
oo
5 5
xoooo
oxooo
ooxoo
oooxo
oooox
```

Sample Output 4

```
..ooo
..ooo
oo.oo
ooo..
ooo..
```

Problem H
Hedge Topiary

Our polygon-shaped bush needs a trim. We would like to cut it down to size so that the remaining leaves of the bush form a new shape of our choosing, with its centre sitting on the top of the stem – represented as the origin $(0, 0)$ in both shapes.

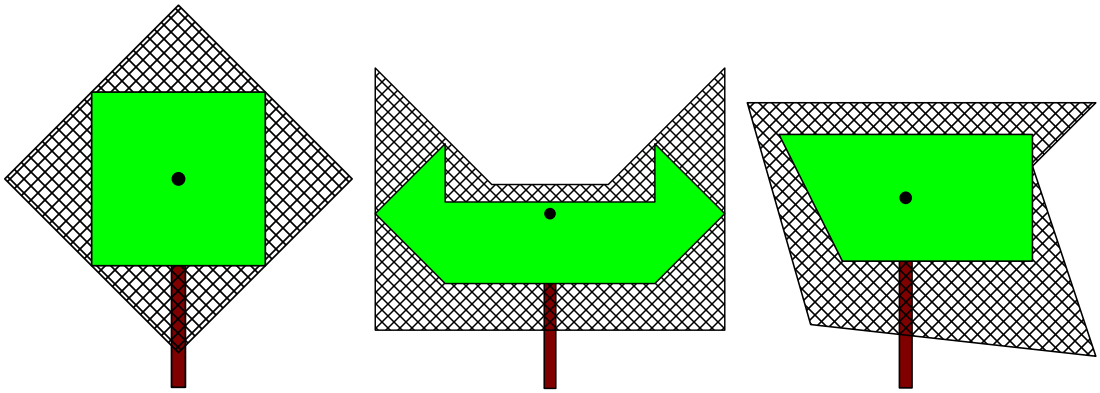


Figure H.1: Bushes cut into beautiful shapes, as given in sample inputs 1, 2, & 3.

The original shape of the bush is a little unusual so it is not obvious how large we can make the new shape without leaving some gaps in the design.

Find out the largest scaling factor that you can apply to the new shape to make it fit into the old shape–meaning that there is no point contained by the re-scaled new shape that was not contained by the old shape as well.

Input

- One line containing the number of coordinates in the new shape, n ($3 \leq n \leq 500$).
- n further lines, the ith of which contains the integer coordinates of the ith vertex in the new shape's polygon, $x_i y_i$ ($-10^4 \leq x, y \leq 10^4$).
- One line containing the number of coordinates in the original shape, m ($3 \leq m \leq 500$).
- n further lines, the ith of which contains the integer coordinates of the ith vertex in the old shape's polygon, $u_i v_i$ ($-10^4 \leq u, v \leq 10^4$).

The old and new shapes are not always convex. However, the origin point is strictly inside (not on the bounds of) both shapes and neither of the shapes self-touch, self-intersect, or repeat any vertices.

Output

Output the maximum amount by which we can scale the first given shape around the origin $(0, 0)$, such that it is fully contained by the bounds of the second given shape. This amount may be any number greater than 0, meaning the shape may also need to become smaller.

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

Sample Input 1

```
4
-1 -1
-1 1
1 1
1 -1
4
-5 0
0 -5
5 0
0 5
```

Sample Output 1

```
2.5
```

Sample Input 2

```
8
-9 1
-9 6
-15 0
-9 -6
9 -6
15 0
9 6
9 1
6
6 -4
6 5
2 1
-2 1
-6 5
-6 -4
```

Sample Output 2

```
0.4
```

Sample Input 3

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

Sample Output 3

```
2
```

Problem I
Inconsistent Patterns

The Simpson's Paradox is a phenomenon in statistics where a trend or pattern that appears in different groups of data is inconsistent (disappears or even reverses) with what we see when the groups are combined. It is named after the British statistician Edward H. Simpson, who described it in 1951, although similar observations had been made earlier.

For example, let assume that two teams have been training for the UKIEPC 2024 and have the following statistics for the graph and geometry problems:

- Team X has solved 81 out of 87 graph problems (success rate of approx 93%), and 192 out of 263 geometry (73%). Total is 273 out of 350 problems (78%).
- Team Y has solved 234 out of 270 graph problems (87%), and 55 out of 80 geometry (69%). Total is 289 out of 350 (83%).

If we look per category — team X has higher success rate in both categories, but when looking in combination, the pattern reverses, and team Y appears to have higher success rate.

In this problem you are to construct an example of the dataset illustrating the Simpson's paradox. More specifically, let us assume (similarly to the example above) that there are two teams who have been solving problems of N categories and the total number of problems solved by each of the teams is M. Let us denote the number of the problems in i-th category solved by Team X as a_i, attempted — by b_i. Similarly, let us define c_i as the number of problems solved by Team Y in the i-th category and d_i as the number of problems attempted.

You are to find such a_i, b_i, c_i and d_i that:

- $\sum b_i = \sum d_i = M$
- $a_i \le b_i$ for all i from 1 to N
- $c_i \le d_i$ for all i from 1 to N
- $a_i, b_i, c_i, d_i > 0$ for all i from 1 to N
- $\frac{a_i}{b_i} > \frac{c_i}{d_i}$ for all i from 1 to N
- $\frac{\sum a_i}{\sum b_i} < \frac{\sum c_i}{\sum d_i}$

Input

Input file contains two integer numbers N and M ($2 \le N \le 10000$, $4 * N \le M \le 10^5$).

Output

Output N lines — i-th of them should contain four positive integer numbers a_i, b_i, c_i, d_i, describing the dataset. Input data is selected in such a way that the solution exists.

Sample Input 1	Sample Output 1
2 350	81 87 234 270 192 263 55 80

Problem J
Jabber Network

Dave, an old Computer Science professor, still maintains a local community computer network even after retirement. Each community member has a computer with three networking cards, and some of these cards may be connected by a cable. They form a connected network, and, following a long resource-saving tradition, the number of cables is kept to the minimum possible.

The habits of all the community members are quite stable: for every two computers the number of packets per second between them is known exactlyc. However, the network was first assembled a long time ago, so the connections are not necessarily be optimal any more. For two computers numbered i and j we define d_{ij} the shortest path between them, measured in the number of cables, and c_{ij} the number of packets per second that should be transferred from i to j. The *commutation stress* is defined to be the sum of $c_{ij} \cdot d_{ij}$ for all $i < j$, and one would like to minimise it.

Dave realised that it is finally the time to upgrade the cables — after all, they do degrade with time. He wants to take this opportunity to also optimise the network, such that the *commutation stress* becomes smaller. However, he is no longer as quick as in his youth, and his friends may get dissatisfied if too much disruption happens at once. So he decided that he will perform the upgrade using the following scenario. For each of the old cables, he will do the following:

1. Remove the old cable.
2. Connect the network back using a new cable, choosing the computers to connect in such a way that the resulting *commutation stress* is minimum possible.
3. If there are many ways to do this, break ties by choosing the computers with the smallest numbers: if (u_1, u_2) and (v_1, v_2) result in the same *commutation stress*, but $u_1 < v_1$ (or $u_1 = v_1$ and $u_2 < v_2$), then (u_1, u_2) should be chosen.

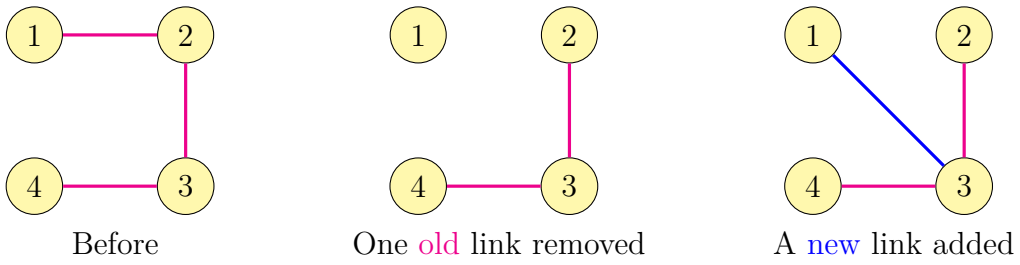


Figure J.1: A single reconnection operation (the first one in the sample input)

Note that, since each computer only has three network cards, Dave cannot connect two arbitrary computers on the second step: if one of them is already connected to three other computers, it is impossible to connect it to yet another computer. Fortunately, it is not hard to show that it is always possible to find two computers to connect: for instance, Dave can choose the two just-disconnected computers.

Unfortunately, the task appeared to be more difficult than it seemed initially. Could you help Dave?

Input

The first line of the input file contains an integer n ($2 \leq n \leq 2 \cdot 10^3$), the number of computers in the network.

The following $n-1$ lines contain two integers each: a_i, b_i, where ($1 \leq a_i < b_i \leq n$) are the numbers of the computers initially connected by an old cable number i. The cables are to be removed and replaced in the order they are given in the input file. It is guaranteed that it is possible to reach any computer from any other computer using old cables (that is, the network is initially connected), and that no computer is connected with more than three other computers.

The next line contains an integer d ($2 \leq d \leq 10^4$), the number of computer pairs that are known to transmit data to each other.

The following d lines contain three integers each: s_i, t_i and d_i, where s_i and t_i are the numbers of the computers which transmit data to each other ($1 \leq s_i < t_i \leq n$), and d_i ($1 \leq d_i \leq 10^9$) is the number of packets per second to be transmitted.

Output

Output $n-1$ lines containing two integers each: x_i, y_i, where ($1 \leq x_i < y_i \leq n$), should be the numbers of the computers connected by a new cable at step i.

Note that, due to the tie-breaking rule detailed above, the correct output is unique.

Sample Input 1	Sample Output 1
4 1 2 2 3 3 4 6 1 2 1 1 3 10 1 4 1 2 3 10 2 4 1 3 4 10	1 3 2 3 3 4

Problem K
Knitting

You are knitting a scarf in a striped pattern, having made strong progress on the first few stripes. You have multiple colours available to continue creating the pattern and would very much like that no colour appears too often – more specifically, that the colour appearing most often still appears as few times as possible.

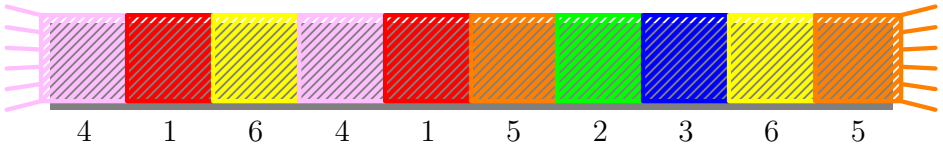


Figure K.1: A knitted solution to sample input 1. No colour is used more than twice, nor is any colour repeated within 3 consecutive stripes.

Please extend the given design to fashion a nice scarf.

Input

- One line containing the number of stripes to create, n, the number of colours available, k, and the minimum spacing between stripes of the same colour, p ($1 \leq n, k, p \leq 10^5$).
- One line containing the number of stripes already knitted, m ($1 \leq m \leq n$).
- One line containing m integers, the colours of the stripes s_i ($1 \leq s \leq k$).

The colours of the stripes are represented as integers in $[1 \ldots k]$.

Output

Output any stripe pattern starting with $s_{1 \ldots m}$ that does not repeat any of the colours too soon and uses the most-frequent colour as few times as possible.

If it is not possible to create a scarf from the parameters, output `impossible` instead.

Sample Input 1	Sample Output 1

`10 6 3`
`4`
`4 1 6 4`
	`4 1 6 4 1 5 2 3 6 5`

Sample Input 2	Sample Output 2

`5 2 3`
`2`
`1 2`
	`impossible`

Sample Input 3

```
2 2 3
1
2
```

Sample Output 3

```
2 1
```

Sample Input 4

```
10 26 5
4
8 3 16 3
```

Sample Output 4

```
impossible
```

Problem L
Leg Day

You have been given a new training plan for the month, consisting of days focusing on legs or arms interspersed with rest days. This training plan will be repeated for as any times as necessary to get to the end of the month.

Each day of the training plan either contains the word ”rest”, in which case it is a rest day, or if not may still contain ”leg”, in which case it is a leg day, or contains neither, meaning that of course it is an arm day.

Produce a motivational 31-day calendar, starting on a Monday, showing what types of exercises you will do on each day.

Input

- One line containing the number of exercises, n ($1 \leq n \leq 31$).
- n further lines, the ith of which contains the name of the ith exercise as between 1 and 50 lowercase Latin characters.

Output

Output 5 rows of UTF-8 text, each containing:

- the **week number** (from 1 to 5)
- Up to 7 pictographs representing the 31 days of the training plan, using glyphs from the Unicode “Supplementary Multilingual Plane” to illustrate the exercises.

You may use any appropriate character as long as the name is a faithful illustration of the exercise, according to the Unicode 17.0 specification. This will be judged as follows:

- The character must be printable
- For leg days, the name of the character must include “leg”
- For arm days, the name of the character must include “arm” or “biceps”
- For rest days, the name of the character must include “face”

For good training results, consistency is key: if you use a character to illustrate a type of activity once, you must always use it to represent that type of activity, and no other type of activity.

Sample Input 1	Sample Output 1
4 legcurls armgains restarms bicepsprints	1 🦵💪😎💪🦵💪😎 2 💪🦵💪😎💪🦵💪 3 😎💪🦵💪😎💪🦵 4 💪😎💪🦵💪😎💪 5 🦵💪😎

Sample Input 2	Sample Output 2
1 workhardplayhardresthard	1 😴😴😴😴😴😴😴 2 😴😴😴😴😴😴😴 3 😴😴😴😴😴😴😴 4 😴😴😴😴😴😴😴 5 😴😴😴

Sample Input 3	Sample Output 3
7 overestimate wrestling crestfallen pharmaceutic forestry elegantrestaurant delegation	1 🤖🤖🤖🦾🤖🤖🦿 2 🤖🤖🤖🦾🤖🤖🦿 3 🤖🤖🤖🦾🤖🤖🦿 4 🤖🤖🤖🦾🤖🤖🦿 5 🤖🤖🤖