Problem A. Admissible Map

Time limit:	3 seconds
Memory limit:	512 megabytes

A *map* is a matrix consisting of symbols from the set of ‘`U`’, ‘`L`’, ‘`D`’, and ‘`R`’.

A *map graph* of a map matrix a is a directed graph with $n \cdot m$ vertices numbered as (i, j) $(1 \le i \le n; 1 \le j \le m)$, where n is the number of rows in the matrix, m is the number of columns in the matrix. The graph has $n \cdot m$ directed edges $(i, j) \to (i + di_{a_{i,j}}, j + dj_{a_{i,j}})$, where $(di_U, dj_U) = (-1, 0)$; $(di_L, dj_L) = (0, -1)$; $(di_D, dj_D) = (1, 0)$; $(di_R, dj_R) = (0, 1)$. A map graph is *valid* when all edges point to valid vertices in the graph.

An *admissible map* is a map such that its map graph is valid and consists of a set of cycles.

A *description* of a map a is a concatenation of all rows of the map — a string $a_{1,1}a_{1,2}\ldots a_{1,m}a_{2,1}\ldots a_{n,m}$.

You are given a string s. Your task is to find how many substrings of this string can constitute a description of some admissible map.

A *substring* of a string $s_1s_2\ldots s_l$ of length l is defined by a pair of indices p and q $(1 \le p \le q \le l)$ and is equal to $s_ps_{p+1}\ldots s_q$. Two substrings of s are considered different when the pair of their indices (p, q) differs, even if they represent the same resulting string.

Input

In the only input line, there is a string s, consisting of at least one and at most 20 000 symbols ‘`U`’, ‘`L`’, ‘`D`’, or ‘`R`’.

Output

Output one integer — the number of substrings of s that constitute a description of some admissible map.

Examples

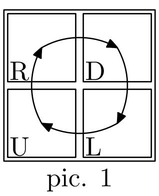
standard input	standard output
`RDUL`	`2`
`RDRU`	`0`
`RLRLRL`	`6`

Note

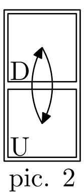
In the first example, there are two substrings that can constitute a description of an admissible map — “`RDUL`” as a matrix of size 2×2 (pic. 1) and “`DU`” as a matrix of size 2×1 (pic. 2).

In the second example, no substring can constitute a description of an admissible map. E. g. if we try to look at the string “`RDRU`” as a matrix of size 2×2, we can find out that the resulting graph is not a set of cycles (pic. 3).

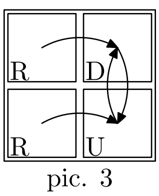
In the third example, three substrings “`RL`”, two substrings “`RLRL`” and one substring “`RLRLRL`” can constitute an admissible map, some of them in multiple ways. E. g. here are two illustrations of substring “`RLRLRL`” as matrices of size 3×2 (pic. 4) and 1×6 (pic. 5).



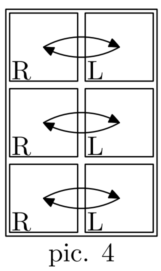
pic. 1



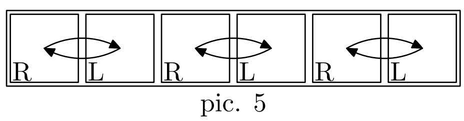
pic. 2



pic. 3



pic. 4



pic. 5

Problem B. Budget Distribution

Time limit: 3 seconds
Memory limit: 512 megabytes

Distributing budgeted money with limited resources and many constraints is a hard problem. A *budget plan* consists of t topics; i-th topic consists of n_i items. For each topic, the *optimal relative money distribution* is known. The optimal relative distribution for the topic i is a list of real numbers $p_{i,j}$, where $\sum_{j=1}^{n_i} p_{i,j} = 1$.

Let's denote the amount of money assigned to j-th item of the topic i as $c_{i,j}$; the total amount of money for the topic is $C_i = \sum_{j=1}^{n_i} c_{i,j}$. A *non-optimality* of the plan for the topic i is defined as $\sum_{j=1}^{n_i} \left|\frac{c_{i,j}}{C_i} - p_{i,j}\right|$. Informally, the non-optimality is the total difference between the optimal and the actual ratios of money assigned to all the items in the topic. The *total plan non-optimality* is the sum of non-optimalities of all t topics. Your task is to minimize the total plan non-optimality.

However, the exact amount of money available is not known yet. j-th item of i-th topic already has $\hat{c}_{i,j}$ dollars assigned to it and they cannot be taken back. Also, there are q possible values of the extra unassigned amounts of money available x_k. For each of them, you need to calculate the minimal possible total non-optimality among all ways to distribute this extra money. You don't need to assign an integer amount of money to an item, any real number is possible, but all the extra money must be distributed among all the items in addition to $\hat{c}_{i,j}$ already assigned. Formally, for each value of extra money x_k you'll need to find its distribution $d_{i,j}$ such that $d_{i,j} \ge 0$ and $\sum_{i=1}^{t}\sum_{j=1}^{n_i} d_{i,j} = x_k$, giving the resulting budget assignments $c_{i,j} = \hat{c}_{i,j} + d_{i,j}$ that minimize the total plan non-optimality.

Input

The first line contains two integers t ($1 \le t \le 5 \cdot 10^4$) and q ($1 \le q \le 3 \cdot 10^5$) — the number of topics in the budget and the number of possible amounts of extra money.

The next t lines contain descriptions of topics. Each line starts with an integer n_i ($2 \le n_i \le 5$) — the number of items in i-th topic; it is followed by n_i integers $\hat{c}_{i,j}$ ($0 \le \hat{c}_{i,j} \le 10^5$; for any i, at least one of $\hat{c}_{i,j} > 0$) — the amount of money already assigned to j-th item in i-th topic; they are followed by n_i integers $p'_{i,j}$ ($1 \le p'_{i,j} \le 1000$) — they determine the values of $p_{i,j}$ as $p_{i,j} = p'_{i,j} / \sum_{j=1}^{n_i} p'_{i,j}$ with $\sum_{j=1}^{n_i} p_{i,j} = 1$.

The next line contains q integers x_k ($0 \le x_k \le 10^{12}$) — k-th possible amount of extra money.

Output

Output q real numbers — the minimal possible non-optimality for the corresponding amount of extra money x_k. An absolute or a relative error of the answer must not exceed 10^{-6}.

Examples

standard input	standard output
1 5 3 1 7 10 700 400 100 0 2 10 50 102	1.0555555555555556 0.8666666666666667 0.5476190476190478 0.12745098039215708 0.0
2 5 3 10 70 100 700 400 100 3 10 30 100 700 400 100 2 10 50 70 110	2.2967032967032974 2.216776340655188 1.8690167362600323 1.7301587301587305 1.5271317829457367

Problem C. Connect the Points

Time limit: 3 seconds
Memory limit: 512 megabytes

You are given three points on a plane. You should choose some segments on the plane that are parallel to coordinate axes, so that all three points become connected. The total length of the chosen segments should be the minimal possible.

Two points a and b are considered connected if there is a sequence of points $p_0 = a, p_1, \ldots, p_k = b$ such that points p_i and p_{i+1} lie on the same segment.

Input

The input consists of three lines describing three points. Each line contains two integers x and y separated by a space — the coordinates of the point ($-10^9 \le x, y \le 10^9$). The points are pairwise distinct.

Output

On the first line output n — the number of segments, at most 100.

The next n lines should contain descriptions of segments. Output four integers x_1, y_1, x_2, y_2 on a line — the coordinates of the endpoints of the corresponding segment ($-10^9 \le x_1, y_1, x_2, y_2 \le 10^9$). Each segment should be either horizontal or vertical.

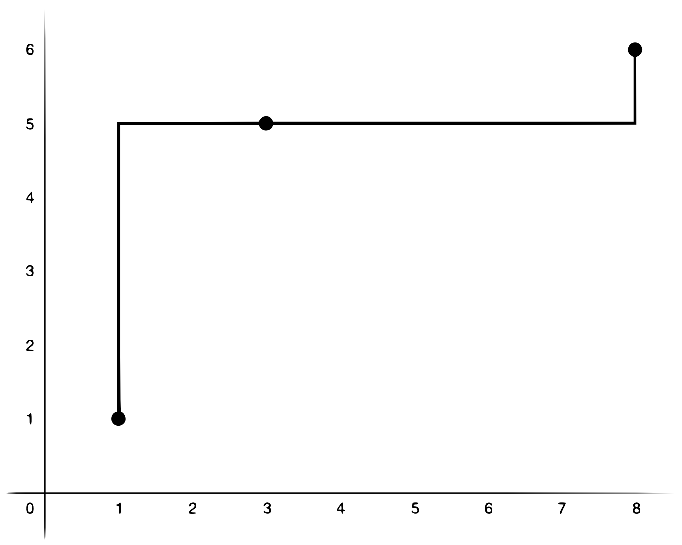
It is guaranteed that the solution with the given constraints exists.

Example

standard input	standard output
1 1 3 5 8 6	3 1 1 1 5 1 5 8 5 8 5 8 6

Note

The points and the segments from the example are shown below.



Problem D. Deletive Editing

Time limit: 3 seconds
Memory limit: 512 megabytes

Daisy loves playing games with words. Recently, she has been playing the following Deletive Editing word game with Daniel.

Daisy picks a word, for example, "`DETERMINED`". On each game turn, Daniel calls out a letter, for example, '`E`', and Daisy removes **the first occurrence** of this letter from the word, getting "`DTERMINED`". On the next turn, Daniel calls out a letter again, for example, '`D`', and Daisy removes its first occurrence, getting "`TERMINED`". They continue with '`I`', getting "`TERMNED`", with '`N`', getting "`TERMED`", and with '`D`', getting "`TERME`". Now, if Daniel calls out the letter '`E`', Daisy gets "`TRME`", but there is no way she can get the word "`TERM`" if they start playing with the word "`DETERMINED`".

Daisy is curious if she can get the final word of her choice, starting from the given initial word, by playing this game for zero or more turns. Your task it help her to figure this out.

Input

The first line of the input contains an integer n — the number of test cases ($1 \le n \le 10\,000$). The following n lines contain test cases.

Each test case consists of two words s and t separated by a space. Each word consists of at least one and at most 30 uppercase English letters; s is the Daisy's initial word for the game; t is the final word that Daisy would like to get at the end of the game.

Output

Output n lines to the output — a single line for each test case. Output "`YES`" if it is possible for Daisy to get from the initial word s to the final word t by playing the Deletive Editing game. Output "`NO`" otherwise.

Example

standard input	standard output
6 DETERMINED TRME DETERMINED TERM PSEUDOPSEUDOHYPOPARATHYROIDISM PEPA DEINSTITUTIONALIZATION DONATION CONTEST CODE SOLUTION SOLUTION	YES NO NO YES NO YES

Problem E. Even Split

Time limit:	3 seconds
Memory limit:	512 megabytes

A revolution has recently happened in Segmentland. The new government is committed to equality, and they hired you to help with land redistribution in the country.

Segmentland is a segment of length l kilometers, with the capital in one of its ends. There are n citizens in Segmentland, the home of i-th citizen is located at the point a_i kilometers from the capital. No two homes are located at the same point. Each citizen should receive a segment of positive length with ends at integer distances from the capital that contains her home. The union of these segments should be the whole of Segmentland, and they should not have common points besides their ends. To ensure equality, the difference between the lengths of the longest and the shortest segments should be as small as possible.

Input

The first line of the input contains two integers l and n ($2 \le l \le 10^9; 1 \le n \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 < a_1 < a_2 < \cdots < a_n < l$).

Output

Output n pairs of numbers s_i, f_i ($0 \le s_i < f_i \le l$), one pair per line. The pair on i-th line denotes the ends of the $[s_i, f_i]$ segment that i-th citizen receives.

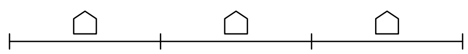
If there are many possible arrangements with the same difference between the lengths of the longest and the shortest segments, you can output any of them.

Examples

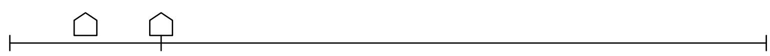
standard input	standard output
6 3 1 3 5	0 2 2 4 4 6
10 2 1 2	0 2 2 10

Note

In the first example, it is possible to make all segments equal. *Viva la revolucion!*



In the second example, citizens live close to the capital, so the length of the shortest segment is 2 and the length of the longest segment is 8.



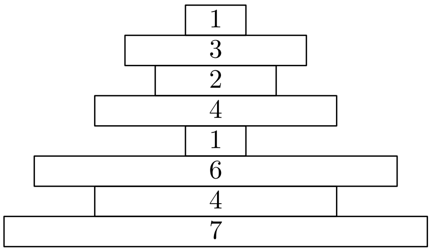
Problem F. Fancy Stack

Time limit:	3 seconds
Memory limit:	512 megabytes

Little Fiona has a collection of n blocks of various sizes $a_1, a_2, \ldots, a_n$, where n is even. Some of the blocks can be equal in size. She would like to put all these blocks one onto another to form a *fancy* stack.

Let $b_1, b_2, \ldots, b_n$ be the sizes of blocks in the stack from top to bottom. Since Fiona is using all her blocks, $b_1, b_2, \ldots, b_n$ must be a permutation of $a_1, a_2, \ldots, a_n$. Fiona thinks the stack is *fancy* if both of the following conditions are satisfied:

- The second block is strictly bigger than the first one, and then each block is alternately strictly smaller or strictly bigger than the previous one. Formally, $b_1 < b_2 > b_3 < b_4 > \ldots > b_{n-1} < b_n$.
- The sizes of the blocks on even positions are strictly increasing. Formally, $b_2 < b_4 < b_6 < \ldots < b_n$ (remember that n is even).



Two stacks are considered different if their corresponding sequences $b_1, b_2, \ldots, b_n$ differ in at least one position.

Fiona wants to know how many different fancy stacks she can build with all of her blocks. Since large numbers scare Fiona, find this number modulo 998 244 353.

Input

Each input contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 2500$). Description of the test cases follows.

The first line of each test case contains a single integer n — the number of blocks at Fiona's disposal ($2 \le n \le 5000$; n is even). The second line contains n integers $a_1, a_2, \ldots, a_n$ — the sizes of the blocks in non-decreasing order ($1 \le a_1 \le a_2 \le \cdots \le a_n \le n$).

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

For each test case, print the number of ways to build a fancy stack, modulo 998 244 353.

Example

standard input	standard output
2 4 1 2 3 4 8 1 1 2 3 4 4 6 7	2 4

Problem G. Global Warming

Time limit:	3 seconds
Memory limit:	512 megabytes

You are developing a new computer game. Let's consider an island in the middle of the ocean in a three-dimensional space with z-axis pointing upwards. The surface of the ocean is a horizontal plane with $z = 0$. The island is a polyhedron of a special form.

You are given n points (x_i, y_i, z_i); there are some edges between them. If we look at the island from above or if we discard z coordinate of each point, points and edges form a planar connected graph. Every face of this planar graph, except the external one, is a nondegenerate triangle. Every edge of the graph belongs to at least one internal face. All points that lie on the edge of the external face of the graph have z coordinates equal to zero. Other points may have arbitrary non-negative z coordinates. Every face of the planar graph corresponds to the face of the polyhedron with the same vertices.

Due to global warming, the level of the ocean is increasing and floods the island. Your task is to compute various global warming scenarios for the game.

In each scenario, the level of the ocean increased to the height h, so that the surface of the ocean is a plane $z = h$. Parts of the island that are **below or at the height** h are now *flooded*, even though some parts may be not reachable from the ocean by water (see the illustration for the second example). In a scenario, a player is standing in p-th point. You shall compute the area of the surface of the part of the island where the player is standing, or determine that the player is standing at or below the water level and has drowned.

Formally, we say that two points on the surface of the island belong to the same part if a player can move between them walking by the surface of the island staying strictly higher than ocean level. Note that you should find the area of the surface of the island itself, not the area of its projection on a horizontal plane.

Input

The first line contains a single integer n — the number of points ($1 \le n \le 10^5$).

Each of the next n lines contains three integers x_i, y_i, and z_i — the coordinates of i-th point ($-10^6 \le x_i, y_i \le 10^6$; $0 \le z_i \le 10^6$).

The next line contains a single integer m — the number of internal faces of the planar graph ($1 \le m \le 10^5$). They are also the faces of the island's polyhedron.

Each of the next m lines contains three integers a_i, b_i, and c_i — the indices of three points that are vertices of i-th internal face ($1 \le a_i, b_i, c_i \le n$).

It is guaranteed that if z coordinate is discarded, then the resulting graph is a connected and planar graph. All its faces, except the external one, are nondegenerate triangles. All points that lie on the edge of the external face of the planar graph have z coordinate equal to zero.

The next line contains an integer q — the number of global warming scenarios to compute ($1 \le q \le 10^5$).

Each of the next q lines contains two integers h_i and p_i — the level of the ocean and the index of the point where the player is standing respectively ($0 \le h_i \le 10^6$; $1 \le p_i \le n$).

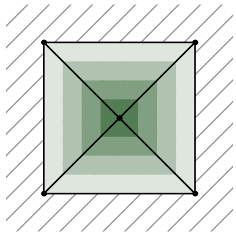
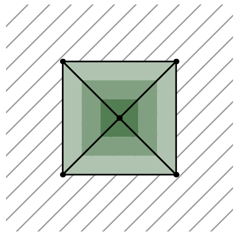
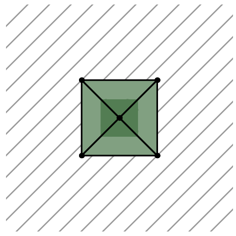
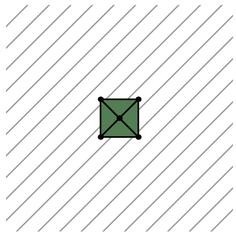
Output

For every scenario output a single real number — the area of the surface of part of the island where the player is standing. If the player's position is flooded by water, output -1 instead.

The answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

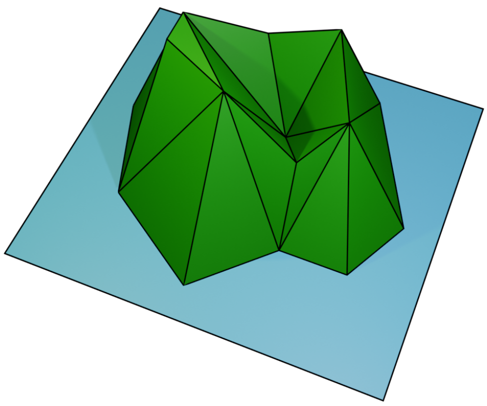
Examples

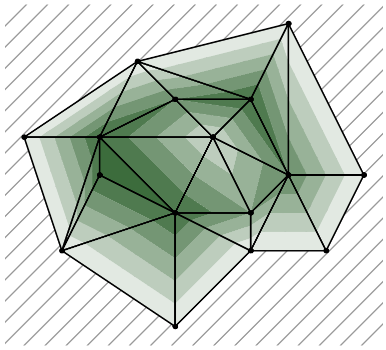
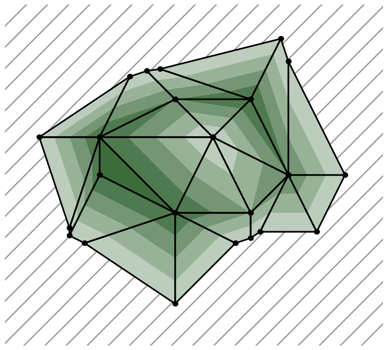
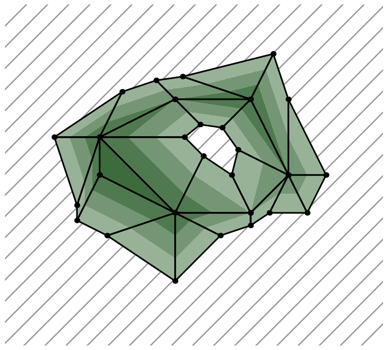
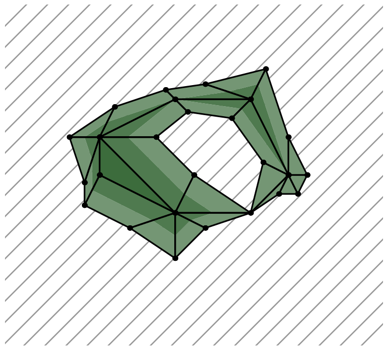
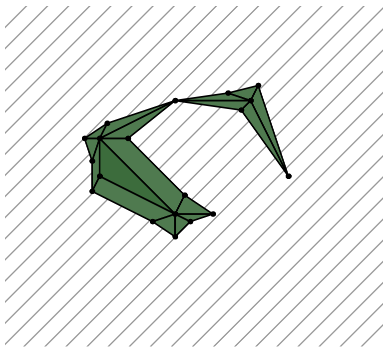
The illustrations of the examples are views of the island from the above. The ocean is hatched. The island is drawn with contour lines, with higher parts in darker colors.

standard input	standard output	illustration
5 0 0 0 2 0 0 2 2 0 0 2 0 1 1 4 4 1 2 5 2 3 5 3 4 5 4 1 5 7 0 1 0 5 1 5 2 5 3 5 4 5 5 5	-1 16.492422502470642 9.276987657639736 4.123105625617661 1.030776406404415 -1 -1	 Ocean level $z = 0$  Ocean level $z = 1$  Ocean level $z = 2$  Ocean level $z = 3$

See the second example on the next page.

The island in the second example looks as follows.



<table>
<tr><th>standard input</th><th>standard output</th><th>illustration</th></tr>
<tr><td>16
0 5 0
1 2 0
2 5 5
3 7 0
4 0 0
4 3 5
5 5 1
6 2 0
6 6 5
7 4 4
7 8 0
8 2 0
9 4 0
4 6 4
6 3 3
2 4 5
22
11 10 9
12 8 10
2 6 5
9 10 7
8 15 6
16 3 6
15 6 7
7 3 14
8 10 15
11 13 10
16 6 2
12 10 13
10 7 15
16 3 2
3 4 1
14 7 9
11 9 4
3 6 7
5 6 8
14 4 3
3 1 2
9 4 14
7
0 7
1 7
1 16
2 10
3 9
4 16
5 16</td><td>120.483405354306325
-1
93.929895222484783
68.181919663536940
40.918561474148331
11.067441790921070
-1</td><td>
Ocean level $z = 0$

Ocean level $z = 1$

Ocean level $z = 2$

Ocean level $z = 3$

Ocean level $z = 4$</td></tr>
</table>

Problem H. Heroes of Might

Time limit:	3 seconds
Memory limit:	512 megabytes

Recently, Hellen played her favorite game "Heroes of Might". She had a hero with only one Rust dragon, which was attacked by another hero with a lot of peasants. Another hero had n groups of peasants, i-th of them had a_i peasants in it. Unfortunately, Hellen lost that battle, but now she is wondering how big the health of the Rust dragon should be to win against such a big army of peasants?

Let's discuss how the battle goes. Initially, the Rust dragon has h_d health points, and each peasant has h_p health points. So i-th group of peasants has a total of $H = h_p \cdot a_i$ health points at the start of the battle. The battle consists of several rounds. In each round, two things happen:

- First, **the dragon chooses one group of peasants and attacks it**. The health of that group is decreased by the dragon's damage rating d. If the group has zero or less health points, it is destroyed and is removed from the game.
- Second, **each one of the peasant groups attacks the dragon**. A group with the total current health H has $\lceil \frac{H}{h_p} \rceil$ peasants still alive and each of them decreases the dragon's health by one.

If the dragon's health becomes zero or less at any point, it dies and Hellen loses. If all peasant groups are destroyed, Hellen wins the battle.

You need to determine the smallest possible h_d, which could make Hellen win if she chooses targets on each turn optimally.

Input

The first line of the input contains an integer t ($1 \le t \le 1000$) — the number of test cases you need to solve.

Each of the test cases is described by two lines. The first line contains three numbers n ($1 \le n \le 1000$), d ($1 \le d \le 10^9$), and h_p ($1 \le h_p \le 10^9$) — the number of peasant groups, the dragon's damage rating, and the health of each peasant. The second line contains n numbers a_i ($1 \le a_i \le 10^9; h_p \cdot \sum a_i \le 10^9$) — the number of peasants in each group.

The sum of n over all test cases does not exceed 1000.

Output

For each test case, output one number — the smallest amount of health h_d that the dragon should have for Hellen to win the battle. If the dragon is never attacked by a peasant, it should still have positive health, so output 1 in this case.

Example

standard input	standard output
4 1 15 10 4 1 10 1 10 2 15 10 4 5 2 11 15 10 17	5 1 26 504

Note

In the third test case, the optimal Hellen's strategy leads to the following battle. At the start, the dragon has $h_d = 26$ health points, and two groups of peasants have $H_1 = 4 \cdot 10$ and $H_2 = 5 \cdot 10$ health points. We'll denote them as $H_1 = 40(4)$ and $H_2 = 50(5)$, placing the value of $\lceil \frac{H}{h_p} \rceil$ in the brackets.

$h_d = 26$, $H_1 = 40(4)$, $H_2 = 50(5)$	Round 1	**The dragon attacks the first group**, dealing 15 damage, leaving $H_1 = 25(3)$.
$h_d = 26$, $H_1 = 25(3)$, $H_2 = 50(5)$		Peasants attack the dragon, dealing $3 + 5$ damage, leaving $h_d = 18$.
$h_d = 18$, $H_1 = 25(3)$, $H_2 = 50(5)$	Round 2	**The dragon attacks the first group**, dealing 15 damage, leaving $H_1 = 10(1)$.
$h_d = 18$, $H_1 = 10(1)$, $H_2 = 50(5)$		Peasants attack the dragon, dealing $1 + 5$ damage, leaving $h_d = 12$.
$h_d = 12$, $H_1 = 10(1)$, $H_2 = 50(5)$	Round 3	**The dragon attacks the second group**, dealing 15 damage, leaving $H_2 = 35(4)$.
$h_d = 12$, $H_1 = 10(1)$, $H_2 = 35(4)$		Peasants attack the dragon, dealing $1 + 4$ damage, leaving $h_d = 7$.
$h_d = 7$, $H_1 = 10(1)$, $H_2 = 35(4)$	Round 4	**The dragon attacks the second group**, dealing 15 damage, leaving $H_2 = 20(2)$.
$h_d = 7$, $H_1 = 10(1)$, $H_2 = 20(2)$		Peasants attack the dragon, dealing $1 + 2$ damage, leaving $h_d = 4$.
$h_d = 4$, $H_1 = 10(1)$, $H_2 = 20(2)$	Round 5	**The dragon attacks the second group**, dealing 15 damage, leaving $H_2 = 5(1)$
$h_d = 4$, $H_1 = 10(1)$, $H_2 = 5(1)$		Peasants attack the dragon, dealing $1 + 1$ damage, leaving $h_d = 2$.
$h_d = 2$, $H_1 = 10(1)$, $H_2 = 5(1)$	Round 6	**The dragon attacks the second group**, destroying it, so it is removed from the game.
$h_d = 2$, $H_1 = 10(1)$		Peasants attack the dragon, dealing 1 damage, leaving $h_d = 1$.
$h_d = 1$, $H_1 = 10(1)$	Round 7	**The dragon attacks the first group**, destroying it, so it is removed from the game.
$h_d = 1$	Game over	The dragon is still alive, Hellen wins.

Problem I. Interactive Treasure Hunt

Time limit: 3 seconds
Memory limit: 512 megabytes

This is an interactive problem.

There is a grid of $n \times m$ cells. Two treasure chests are buried in two different cells of the grid. Your task is to find both of them. You can make two types of operations:

- `DIG` r c: try to find the treasure in the cell (r, c). The interactor will tell you if you found the treasure or not.
- `SCAN` r c: scan from the cell (r, c). The result of this operation is the sum of Manhattan distances from the cell (r, c) to the cells where the treasures are hidden. Manhattan distance from a cell (r_1, c_1) to a cell (r_2, c_2) is calculated as $|r_1 - r_2| + |c_1 - c_2|$.

You need to find the treasures in at most 7 operations. This includes both `DIG` and `SCAN` operations in total. To solve the test you need to call `DIG` operation at least once in both of the cells where the treasures are hidden.

Interaction Protocol

Your program has to process multiple test cases in a single run. First, the testing system writes t — the number of test cases ($1 \le t \le 100$). Then, t test cases should be processed one by one.

In each test case, your program should start by reading the integers n and m ($2 \le n, m \le 16$).

Then, your program can make queries of two types:

- `DIG` r c ($1 \le r \le n$; $1 \le c \le m$). The interactor responds with integer 1, if you found the treasure, and 0 if not. If you try to find the treasure in the same cell multiple times, the result will be 0 since the treasure is already found.
- `SCAN` r c ($1 \le r \le n$; $1 \le c \le m$). The interactor responds with an integer that is the sum of Manhattan distances from the cell (r, c) to the cells where the treasures were hidden. The sum is calculated for both cells with treasures, even if you already found one of them.

After you found both treasures, i. e. you received 1 for two `DIG` operations, your program should continue to the next test case or exit if that test case was the last one.

Example

standard input	standard output
1	
2 3	
	SCAN 1 2
1	
	DIG 1 2
1	
	SCAN 2 2
3	
	DIG 1 1
0	
	DIG 1 3
1	

Problem J. Job Lookup

Time limit: 3 seconds
Memory limit: 512 megabytes

Julia's n friends want to organize a startup in a new country they moved to. They assigned each other numbers from 1 to n according to the jobs they have, from the most front-end tasks to the most back-end ones. They also estimated a matrix c, where $c_{ij} = c_{ji}$ is the average number of messages per month between people doing jobs i and j.

Now they want to make a hierarchy tree. It will be a **binary tree** with each node containing one member of the team. Some member will be selected as a leader of the team and will be contained in the root node. In order for the leader to be able to easily reach any subordinate, for each node v of the tree, the following should apply: all members in its left subtree must have smaller numbers than v, and all members in its right subtree must have larger numbers than v.

After the hierarchy tree is settled, people doing jobs i and j will be communicating via the shortest path in the tree between their nodes. Let's denote the length of this path as d_{ij}. Thus, the cost of their communication is $c_{ij} \cdot d_{ij}$.

Your task is to find a hierarchy tree that minimizes the total cost of communication over all pairs: $\sum_{1 \le i < j \le n} c_{ij} \cdot d_{ij}$.

Input

The first line contains an integer n ($1 \le n \le 200$) – the number of team members organizing a startup.

The next n lines contain n integers each, j-th number in i-th line is c_{ij} — the estimated number of messages per month between team members i and j ($0 \le c_{ij} \le 10^9; c_{ij} = c_{ji}; c_{ii} = 0$).

Output

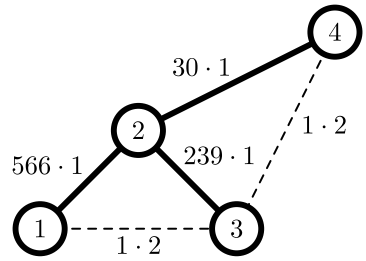
Output a description of a hierarchy tree that minimizes the total cost of communication. To do so, for each team member from 1 to n output the number of the member in its parent node, or 0 for the leader. If there are many optimal trees, output a description of any one of them.

Example

standard input	standard output
4 0 566 1 0 566 0 239 30 1 239 0 1 0 30 1 0	2 4 2 0

Note

The minimal possible total cost is $566 \cdot 1 + 239 \cdot 1 + 30 \cdot 1 + 1 \cdot 2 + 1 \cdot 2 = 839$:



Problem K. Kingdom Partition

Time limit:	3 seconds
Memory limit:	512 megabytes

The King is gone. After the King's rule, all the roads in the Kingdom are run down and need repair. Three of the King's children, Adrian, Beatrice and Cecilia, are dividing the Kingdom between themselves.

Adrian and Beatrice do not like each other and do not plan to maintain any relations between themselves in the future. Cecilia is on good terms with both of them. Moreover, most of the Kingdom's workers support Cecilia, so she has better resources and more opportunity to repair the infrastructure and develop the economy.

Cecilia proposes to partition the Kingdom into three districts: A (for Adrian), B (for Beatrice), and C (for Cecilia), and let Adrian and Beatrice to negotiate and choose any towns they want to be in their districts, and agree on how they want to partition the Kingdom into three districts.

Adrian's castle is located in town a, and Beatrice's one is located in town b. So Adrian and Beatrice want their castles to be located in districts A and B, respectively. Cecilia doesn't have a castle, so district C can consist of no towns.

There is an issue for Adrian and Beatrice. When they choose the towns, they will have to pay for the roads' repair.

The cost to repair the road of length l is $2l$ gold. However, Adrian and Beatrice don't have to bear all the repair costs. The repair cost for the road of length l that they bear depends on what towns it connects:

- For a road between two towns inside district A, Adrian has to pay $2l$ gold;
- For a road between two towns inside district B, Beatrice has to pay $2l$ gold;
- For a road between towns from district A and district C, Adrian has to pay l gold, Cecilia bears the remaining cost;
- For a road between towns from district B and district C, Beatrice has to pay l gold, Cecilia bears the remaining cost.

The roads that connect towns from district A and district B won't be repaired, since Adrian and Beatrice are not planning to use them, so no one pays for them. Cecilia herself will repair the roads that connect the towns inside district C, so Adrian and Beatrice won't bear the cost of their repair either.

Adrian and Beatrice want to minimize the total cost they spend on roads' repair. Find the cheapest way for them to partition the Kingdom into three districts.

Input

The first line contains two integers n and m — the number of towns and the number of roads in the Kingdom ($2 \le n \le 1000$; $0 \le m \le 2000$).

The second line contains two integers that represent town a and town b — the towns that have to be located in district A and district B, respectively ($1 \le a, b \le n$; $a \ne b$).

The following m lines describe the Kingdom roads. The i-th of them consists of three integers u_i, v_i, and l_i representing a road of length l_i between towns u_i and v_i ($1 \le u_i, v_i \le n$; $u_i \ne v_i$; $1 \le l_i \le 10^9$).

Each pair of towns is connected with at most one road.

Output

In the first line output a single integer — the minimum total cost of roads' repair for Adrian and Beatrice.

In the second line output a string consisting of n characters '`A`', '`B`', and '`C`', i-th of the characters representing the district that the i-th town should belong to.

If several cheapest ways to partition the Kingdom exist, print any of them.

Example

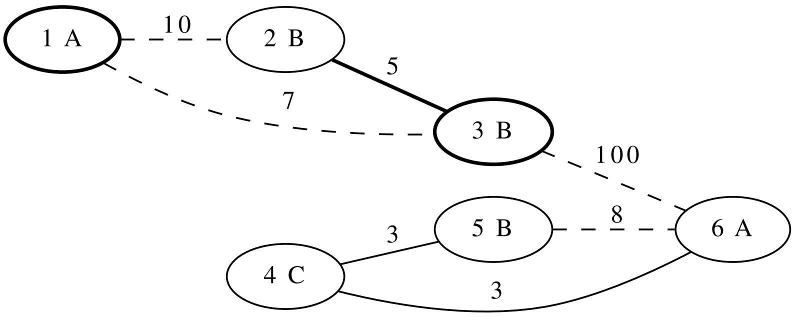
standard input	standard output
6 7 1 3 1 2 10 2 3 5 1 3 7 4 5 3 3 6 100 4 6 3 5 6 8	16 ABBCBA

Note

The following picture illustrates the example. Adrian and Beatrice don't pay for the dashed roads, they pay $2l$ for the bold roads, and l for the solid roads.

So the total cost is $2 \cdot 5 + 3 + 3 = 16$.

The castles of Adrian and Beatrice are located in bold towns.



Problem L. Labyrinth

Time limit:	3 seconds
Memory limit:	512 megabytes

Leslie and Leon entered a labyrinth. The labyrinth consists of n halls and m one-way passages between them. The halls are numbered from 1 to n.

Leslie and Leon start their journey in the hall s. Right away, they quarrel and decide to explore the labyrinth separately. However, they want to meet again at the end of their journey.

To help Leslie and Leon, your task is to find two different paths from the given hall s to some other hall t, such that these two paths do not share halls other than the staring hall s and the ending hall t. The hall t has not been determined yet, so you can choose any of the labyrinth's halls as t except s.

Leslie's and Leon's paths do not have to be the shortest ones, but their paths must be simple, visiting any hall at most once. Also, they cannot visit any common halls except s and t during their journey, even at different times.

Input

The first line contains three integers n, m, and s, where n ($2 \le n \le 2 \cdot 10^5$) is the number of vertices, m ($0 \le m \le 2 \cdot 10^5$) is the number of edges in the labyrinth, and s ($1 \le s \le n$) is the starting hall.

Then m lines with descriptions of passages follow. Each description contains two integers u_i, v_i ($1 \le u_i, v_i \le n$; $u_i \ne v_i$), denoting a passage from the hall u_i to the hall v_i. The passages are one-way. Each tuple (u_i, v_i) is present in the input at most once. The labyrinth can contain cycles and is not necessarily connected in any way.

Output

If it is possible to find the desired two paths, output "`Possible`", otherwise output "`Impossible`".

If the answer exists, output two path descriptions. Each description occupies two lines. The first line of the description contains an integer h ($2 \le h \le n$) — the number of halls in a path, and the second line contains distinct integers $w_1, w_2, \ldots, w_h$ ($w_1 = s$; $1 \le w_j \le n$; $w_h = t$) — the halls in the path in the order of passing. Both paths must end at the same vertex t. The paths must be different, and all intermediate halls in these paths must be distinct.

Examples

standard input	standard output
5 5 1 1 2 2 3 1 4 4 3 3 5	Possible 3 1 2 3 3 1 4 3
5 5 1 1 2 2 3 3 4 2 5 5 4	Impossible
3 3 2 1 2 2 3 3 1	Impossible