Problem A
Hissing Microphone
Problem ID: hissingmicrophone
Time Limit: 1 second

A known problem with some microphones is the "hissing s". That is, sometimes the sound of the letter s is particularly pronounced; it stands out from the rest of the word in an unpleasant way.

Of particular annoyance are words that contain the letter s twice in a row. Words like `amiss`, `kiss`, `mississippi` and even `hiss` itself.

Input

The input contains a single string on a single line. This string consists of only lowercase letters (no spaces) and has between 1 and 30 characters.

Output

Output a single line. If the input string contains two consecutive occurrences of the letter `s`, then output `hiss`. Otherwise, output `no hiss`.

Sample Input	Sample Output
`amiss`	`hiss`

Sample Input	Sample Output
`octopuses`	`no hiss`

Sample Input	Sample Output
`hiss`	`hiss`

This page is intentionally left blank.

Problem B
Open-Pit Mining
Problem ID: openpitmining
Time Limit: 1 second

Open-pit mining is a surface mining technique of extracting rock or minerals from the earth by their removal from an open pit or borrow. Open-pit mines are used when deposits of commercially useful minerals or rocks are found near the surface. Automatic Computer Mining (ACM) is a company that would like to maximize its profits by open-pit mining. ACM has hired you to write a program that will determine the maximum profit it can achieve given the description of a piece of land.

Each piece of land is modelled as a set of blocks of material. Block i has an associated value (v_i), as well as a cost (c_i), to dig that block from the land. Some blocks obstruct or bury other blocks. So for example if block i is obstructed by blocks j and k, then one must first dig up blocks j and k before block i can be dug up. A block can be dug up when it has no other blocks obstructing it.

Input

The first line of input is an integer N ($1 \leq N \leq 200$) which is the number of blocks. These blocks are numbered 1 through N.

Then follow N lines describing these blocks. The ith such line describes block i and starts with two integers v_i, c_i denoting the value and cost of the ith block ($0 \leq v_i, c_i \leq 200$).

Then a third integer $0 \leq m_i \leq N - 1$ on this line describes the number of blocks that block i obstructs. Following that are m_i distinct space separated integers between 1 and N (but excluding i) denoting the label(s) of the blocks that block i obstructs.

You may assume that it is possible to dig up every block for some digging order. The sum of values m_i over all blocks i will be at most 500.

Output

Output a single integer giving the maximum profit that ACM can achieve from the given piece of land.

Sample Input	Sample Output
5 0 3 2 2 3 1 3 2 4 5 4 8 1 4 5 3 0 9 2 0	2

This page is intentionally left blank.

Problem C
Multiplication Game
Problem ID: multiplicationgame
Time Limit: 5 seconds

Alice and Bob are in their class doing drills on multiplication and division. They quickly get bored and instead decide to play a game they invented.

The game starts with a target integer $N \geq 2$, and an integer $M = 1$. Alice and Bob take alternate turns. At each turn, the player chooses a prime divisor p of N, and multiply M by p. If the player's move makes the value of M equal to the target N, the player wins. If $M > N$, the game is a tie.

Assuming that both players play optimally, who (if any) is going to win?

Input

The first line of input contains T ($1 \leq T \leq 10000$), the number of cases to follow. Each of the next T lines describe a case. Each case is specified by N ($2 \leq N \leq 2^{31} - 1$) followed by the name of the player making the first turn. The name is either `Alice` or `Bob`.

Output

For each case, print the name of the winner (`Alice` or `Bob`) assuming optimal play, or `tie` if there is no winner.

Sample Input	Sample Output
10 10 Alice 20 Bob 30 Alice 40 Bob 50 Alice 60 Bob 70 Alice 80 Bob 90 Alice 100 Bob	Bob Bob tie tie Alice tie tie tie tie Alice

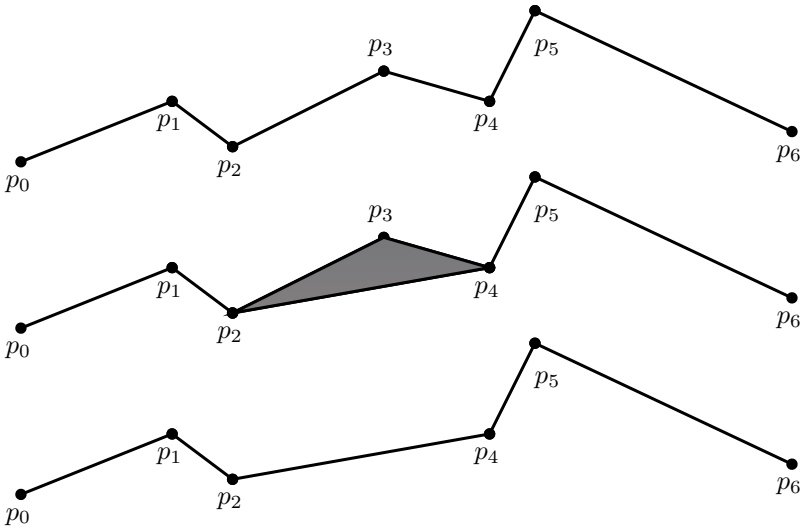
This page is intentionally left blank.

Problem D
Polyline Simplification
Problem ID: simplification
Time Limit: 5 seconds

Mapping applications often represent the boundaries of countries, cities, etc. as polylines, which are connected sequences of line segments. Since fine details have to be shown when the user zooms into the map, these polylines often contain a very large number of segments. When the user zooms out, however, these fine details are not important and it is wasteful to process and draw the polylines with so many segments. In this problem, we consider a particular polyline simplification algorithm designed to approximate the original polyline with a polyline with fewer segments.

A polyline with n segments is described by $n+1$ points $p_0 = (x_0, y_0), \ldots, p_n = (x_n, y_n)$, with the ith line segment being $\overline{p_{i-1}p_i}$. The polyline can be simplified by removing an interior point p_i ($1 \leq i \leq n-1$), so that the line segments $\overline{p_{i-1}p_i}$ and $\overline{p_ip_{i+1}}$ are replaced by the line segment $\overline{p_{i-1}p_{i+1}}$. To select the point to be removed, we examine the area of the triangle formed by p_{i-1}, p_i, and p_{i+1} (the area is 0 if the three points are colinear), and choose the point p_i such that the area of the triangle is smallest. Ties are broken by choosing the point with the lowest index. This can be applied again to the resulting polyline, until the desired number m of line segments is reached.

Consider the example below.



The original polyline is shown at the top. The area of the triangle formed by p_2, p_3, and p_4 is considered (middle), and p_3 is removed if the area is the smallest among all such triangles. The resulting polyline after p_3 is removed is shown at the bottom.

Input

The first line of input contains two integers n ($2 \leq n \leq 200\,000$) and m ($1 \leq m < n$). The next $n+1$ lines specify $p_0, \ldots, p_n$. Each point is given by its x and y coordinates which are integers between -5000 and 5000 inclusive. You may assume that the given points are strictly increasing in lexicographical order. That is, $x_i < x_{i+1}$, or $x_i = x_{i+1}$ and $y_i < y_{i+1}$ for all $0 \leq i < n$.

Output

Print on the kth line the index of the point removed in the kth step of the algorithm described above (use the index in the original polyline).

Sample Input	Sample Output
10 7 0 0 1 10 2 20 25 17 32 19 33 5 40 10 50 13 65 27 75 22 85 17	1 9 6

Problem E
Palindromic Password
Problem ID: palindromicpassword
Time Limit: 3 seconds

The IT department at your school decided to change their password policy. Each password will have to consist of N 6-digit numbers separated by dashes, where N will be determined by the phase of the moon and the weather forecast for the day after it will be generated.

You realized that, if all of the numbers were palindromes (same numbers as the original ones if read backwards), you would have to remember a bunch of 3-digit numbers, which did not sound that bad (at the time).

In order to generate your password of N numbers, you get a list of N randomly generated 6-digit numbers and find the palindromic number closest to them.

Of course, you would like to automate this process...

Input

The first line of the input contains a single positive integer $N \leq 1000$ indicating the number of six-digit numbers in the input. Each of the next N lines contains a six-digit number without leading zeroes.

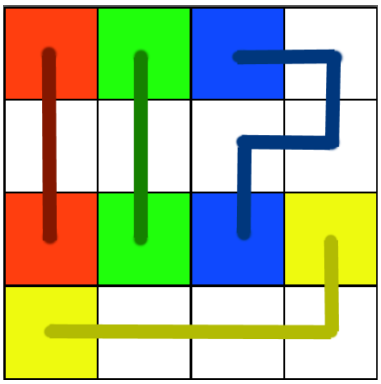
Output

For each six-digit number in the input, output another six-digit number that is closest to it and is also a palindrome. "Closest" in this context means "a number having the smallest absolute difference with the original number". If there are two different numbers satisfying the above condition, output the smaller one of the two. Remember, no leading zeroes.

Sample Input	Sample Output
2 123321 123322	123321 123321

This page is intentionally left blank.

Problem F
Flow Free
Problem ID: flowfree
Time Limit: 2 seconds



Flow Free is a puzzle that is played on a 2D grid of cells, with some cells marked as endpoints of certain colors and the rest of cells being blank.

To solve the puzzle, you have to connect each pair of colored endpoints with a path, following these rules:

- there is a path connecting two points with the same color, and that path also has that color
- all cells in the grid are used and each cell belongs to exactly one path (of the same color as the endpoints of the path)

The rules imply that different colored paths cannot intersect.

The path is defined as a connected series of segments where each segment connects two neighbouring cells. Two cells are neighbours if they share a side (so all segments are either horizontal or vertical). By these definitions and rules above, each colored cell will be an endpoint of exactly one segment and each blank cell will be an endpoint of exactly two segments.

In this problem we will consider only the 4×4 puzzle, with 3 or 4 pairs of colored endpoints given.

Your task is to determine if a given puzzle is solvable or not.

Input

The input consists of 4 lines, each line containing 4 characters. Each character is from the set $\{R, G, B, Y, W\}$ where W denotes the blank cells and the other characters denote endpoints with the specified color. You are guaranteed that there will be exactly 3 or 4 pairs of colored cells. If there are 3 colors in the grid, Y will be omitted.

Output

On a single line output either "`solvable`" or "`not solvable`" (without the quotes).

Sample Input	Sample Output

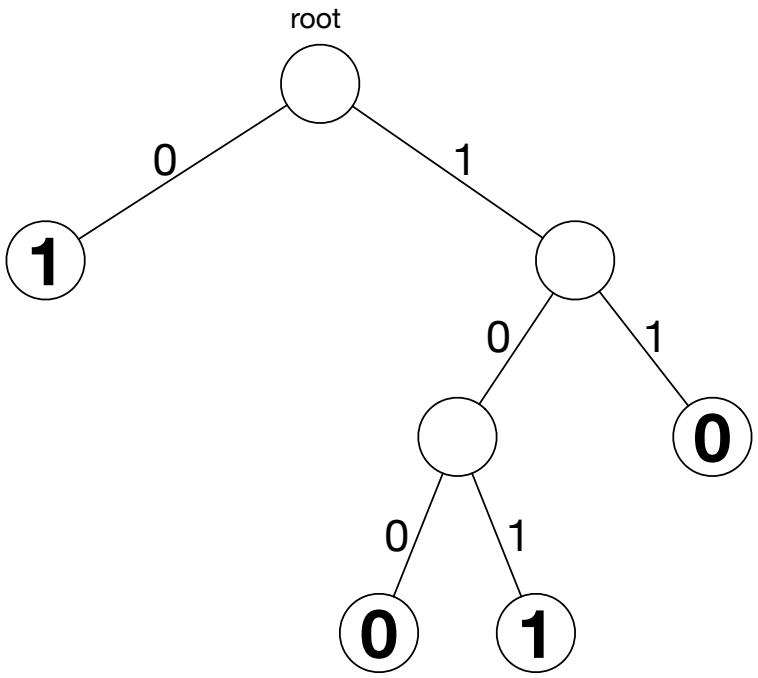
`RGBW`
`WWWW`
`RGBY`
`YWWW`
	`solvable`

Sample Input	Sample Output
RGBW WWWW RGBW YWWY	not solvable

Problem G
Decisions, Decisions
Problem ID: decisions
Time Limit: 2 seconds

Let $x_0, \ldots, x_{n-1}$ denote n boolean variables (i.e., variables taking only values 0 and 1). A *binary decision diagram* (BDD) over these variables is a diagrammatic representation of a boolean function $f(x_0, \ldots, x_{n-1})$ as inputs.

A BDD is a rooted binary tree such that all internal vertices v have precisely two children. The edges connecting an internal vertex v with its children are labelled with a 0 or 1 (exactly one of each). Each leaf vertex is also labelled with a 0 or 1. We note that a BDD may consist of a single vertex, which is considered to be both the root and a leaf vertex.



Given input $(x_0, \ldots, x_{n-1})$, the boolean function represented by the BDD is evaluated as follows.

- **let** v be the root vertex
- **let** $i \leftarrow 0$
- **while** v is not a leaf **do**
  - replace v with the child vertex of v by traversing the edge labelled x_i
  - increase i by 1
- **output** the label of leaf vertex v

Consider the function $f(x_0, x_1, x_2)$ represented by the BDD above. To evaluate $f(1, 0, 1)$, we start from the root, we descend along edges labelled 1, 0, and then 1. We reach a leaf vertex labelled 1, so $f(1, 0, 1) = 1$.

A BDD is *minimal* if there is no way to replace any subtree of an internal vertex of the BDD by a single leaf vertex to get a new BDD defining the same boolean function. The BDD depicted above is minimal. It is a fact that for each boolean function f, there is a unique minimal BDD that represents the boolean function.

In this problem, you are given an n-variable boolean function specified by a list of the 2^n different values the function should take for various inputs. Compute the number of vertices in the minimal BDD representing this function.

Input

The first line of input consists of a single integer $1 \leq n \leq 18$. Then one more line follows that contains 2^n values (either 0 or 1) describing an n-variable boolean function.

We think of these values as being indexed from 0 to $2^n - 1$. The ith such value represents $f(x_0, \ldots, x_{n-1})$ where x_j is the jth least-significant bit of the binary representation of i. In other words, x_j is the coefficient of 2^j in the binary expansion of i.

The third sample input below corresponds to the BDD depicted above.

Output

Output consists of a single integer m that is the number of vertices in the unique minimal BDD representing the boolean function from the input.

Sample Input	Sample Output
2 1 1 0 1	5

Sample Input	Sample Output
2 0 0 0 0	1

Sample Input	Sample Output
3 1 0 1 0 1 1 1 0	7

Problem H
Heart Rate
Problem ID: heartrate
Time Limit: 2 seconds

A common method for determining your own heart rate is to place your index and third finger on your neck to the side of your windpipe. You then count how many beats you feel in a span of 15 seconds, multiply that number by four and that gives you a measure of your heart rate in beats per minute (BPM). This method gives a good estimate, but is not quite accurate. In general, if you measure b beats in p seconds the BPM is calculated as $\frac{60b}{p}$.

For this problem, we assume that all heart rates are constant and do not change. If t is the amount of time (in seconds) between each beat of your heart, we define your Actual Beats Per Minute (ABPM) as $\frac{60}{t}$.

Input

The input starts with an integer N ($1 \leq N \leq 1000$) indicating the number of cases to follow. Each of the next N lines specify one case, consisting of the integer b ($2 \leq b \leq 1000$) as well as p ($0 < p < 1000$) as described above. The value of p is a real number specified to 4 decimal places.

Output

For each case, print on a single line the minimum possible ABPM, the calculated BPM, and the maximum possible ABPM, separated by a space. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-4}.

Sample Input	Sample Output
2 6 5.0000 2 3.1222	60.0000 72.0000 84.0000 19.2172 38.4344 57.6517

This page is intentionally left blank.

Problem I
Initials
Problem ID: initials
Time Limit: 3 seconds

When marking computer science assignments, markers like to mark each student in order of how they are listed in class. The way the class is listed is quite strange though, a class is sorted by concatenating the first name to the last name (for example Harry Potter becomes PotterHarry) and all upper case letters are turned to lower case. Then sorting those strings lexicographically is the order of the class. Each student needs a Unix directory for the marker to put their assignments in.

Since the marker is lazy, he decides to create a directory for each student only using their initials (first letter of last name followed by the first letter of first name). The marker soon realizes that the way Unix sorts the directories, students are no longer sorted in the correct order and there are now duplicate folder names. Oh no! This must be fixed.

For each name, we are allowed to add 0 or more letters from the original name to the initials, except you must always use the first k unchosen letters in the initials. For example, the initials of Harry Potter are PH. Adding 2 letters make it PotH, adding 6 letters makes it PotterHa.

The markers come to you for help. Your job is to write a program which tells the markers the minimum number of letters they need to add to put all the students into correct sorted order, so that all resulting "extended" initials are unique.

For example, in the sample input below, the initials (sorted) would be (HA, HB, ZZ). By adding three letters we get (HaB, HatA, ZZ) which is the correct sorted order.

Input

The first line contains a single integer $2 \leq N \leq 1000$ indicating the number of students. Then follow N lines, each containing two strings: the first and last name of the student. The first and last name each contains at least one character, and they consist of only lowercase and uppercase English alphabetic letters. The combined length of each student name will be at most 80 characters. No two students will have the same concatentated name (as described above).

Output

A single number on a line by itself, the minimum number of letters that need to be added to put all the students in sorted order.

Sample Input	Sample Output
3 Bob Harris Andrea Hat Zanny Zan	3

This page is intentionally left blank.

Problem J
Particle Collision
Problem ID: particlecollision
Time Limit: 1 second

Particle colliders are difficult to build and experiments are costly to run. Before running any real experiments it is better to do a simulation to test out the ideas first. You are required to write a very simple simulator for this problem.

There are only three particles in this system, and all particles are confined to an infinite plane so that they can be modelled as circles. Their locations are specified only by the x_i and y_i coordinates of their centers ($1 \leq i \leq 3$). All three particles have the same radius r, and are initially stationary.

We are given a vector (x_v, y_v) specifying the direction particle 1 will move when the experiment starts. When particle i hits particle j, particle j will start moving in the direction perpendicular to the tangent at the point of the contact, away from particle i. Particle i will cease to exist and be converted to radiation. A moving particle that does not hit another will continue moving indefinitely.

There are a number of possible scenarios your simulator should identify:

1. particle 1 hits particle 2, which in turns hits particle 3;
2. particle 1 hits particle 3, which in turns hits particle 2;
3. particle 1 hits particle 2, which moves indefinitely;
4. particle 1 hits particle 3, which moves indefinitely;
5. particle 1 moves indefinitely.

Input

The input contains four lines. The first three lines each contains two integers x_i and y_i ($|x_i|, |y_i| \leq 1000$), describing particles 1, 2, and 3 in this order. The fourth line contains three integers x_v, y_v, and r ($|x_v|, |y_v| \leq 1000$, $0 < r \leq 50$).

You may assume that no two particles touch or overlap initially, and that the distance between the centers of particles 2 and 3 is greater than $4r$.

Output

Output a single integer giving the number (1–5) identifying the scenarios described above.

Although you should take care of your calculations, it is guaranteed that the outcome would not change if the initial vector (x_v, y_v) is rotated by one degree either way.

Sample Input	Sample Output
0 0 50 45 91 50 42 50 10	1

Sample Input	Sample Output
0 0 50 50 141 50 41 50 10	3

Sample Input	Sample Output
0 0 50 50 131 50 100 50 10	4

Problem K
Frosh Week
Problem ID: froshweek2
Time Limit: 4 seconds

Professor Zac is trying to finish a collection of tasks during the first week at the start of the term. He knows precisely how long each task will take, down to the millisecond. Unfortunately, it is also Frosh Week. Zac's office window has a clear view of the stage where loud music is played. He cannot focus on any task when music is blaring.

The event organizers are also very precise. They supply Zac with intervals of time when music will not be playing. These intervals are specified by their start and end times down to the millisecond.

Each task that Zac completes must be completed in one quiet interval. He cannot pause working on a task when music plays (he loses his train of thought). Interstingly, the lengths of the tasks and quiet intervals are such that it is impossible to finish more than one task per quiet interval!

Given a list of times t_i (in milliseconds) that each task will take and a list of times ℓ_j (in milliseconds) specifying the lengths of the intervals when no music is being played, what is the maximum number of tasks that Zac can complete?

Input

The first line of input contains a pair of integers n and m, where n is the number of tasks and m is the number of time intervals when no music is played. The second line consists of a list of integers $t_1, t_2, \ldots, t_n$ indicating the length of time of each task. The final line consists of a list of times $\ell_1, \ell_2, \ldots, \ell_m$ indicating the length of time of each quiet interval when Zac is at work this week.

You may assume that $1 \leq n, m \leq 200,000$ and $100,000 \leq t_i, \ell_j \leq 199,999$ for each task i and each quiet interval j.

Output

Output consists of a single line containing a single integer indicating the number of tasks that Zac can accomplish from his list during this first week.

Sample Input	Sample Output
5 4 150000 100000 160000 100000 180000 190000 170000 140000 160000	4

Sample Input	Sample Output
4 4 180000 185000 199999 100000 199999 180000 170000 120000	3

Sample Input	Sample Output
3 3 199999 180000 170001 199999 170000 180000	2