A. Waste sorting

1 second, 256 megabytes

"Reduce, reuse, recycle" are the three pillars of modern-day waste reduction. Although recycling is useful, not everybody is familiar with the rules of recycling and what type of waste is suited for what type of recycling bins. As a consequence, some people don't dispose of their waste in the right recycling container. There are n recycling stations in town, each of which has been assigned to a waste of type o. Waste of type o consists of 5 characters. There are also m recycling containers in town, and again, each has been assigned to a waste of type o which consists of 5 characters.

Once they have been filled, the recycling containers have to be brought to a recycling station. The decision of which recycling station is best suited for which recycling container is based upon the similarity between a waste of type o. Each container is assigned to a station whose waste of type o is the most similar to his own.

The similarity between the two types of waste is determined by the number of characters they have in common. An example is type of waste for the terminal is `papir` and container with a waste type of `papin`. The first 5 characters are the same 'papi' and there is a difference in the last character which means that their similarity is $\frac{4}{5}$ which is 80%.

Input

The first line contains n, the number of recycling stations $1 \leq n \leq 100$.

The second line contains n strings of length 5, separated with a whitespace character.

The third line contains m, the number of recycling containers $1 \leq n \leq 100$.

The fourth line contains m strings of length 5, separated by whitespace character.

Output

For each of the m containers, print which recycling station it is best suited for.

input

```
2
papir metal
3
papin metal papek
```

output

```
1
2
1
```

B. Mysterious DJ

1 second, 256 megabytes

Every day at the Stem Games in Rovinj, one of the artists is performing and entertaining the crowd. The time has come for the main star - DJ Array who will be performing one of his greatest hits. Unfortunately, DJ has a problem with his most popular mix where the starting (fundamental) frequencies have been lost and only 3 frequencies H_i remained. These 3 frequencies are consecutive, meaning that they appear next to each other in the original mix. It is also guaranteed that the three frequencies are given in their original order such that for a given frequency f_i, the other two frequencies are f_{i+1} and f_{i+2}, respectively. Their position i in the original mix is unknown.

In addition to having the 3 remaining frequencies, DJ Array can also remember S, the sum of the first 5 frequencies from the original mix.

H_i - an array of consecutive frequencies (separated by whitespace) S - a sum of the first 5 frequencies

Input

The first line contains the three remaining frequencies: $1 \leq f_i < f_{i+1} < f_{i+2} \leq 10^9$, each separated by a whitespace

The second line contains S, $1 \leq 1S \leq 10^6$, a single number representing the sum of the first five frequencies from the original mix.

Output

The output is a single number representing the 1st (fundamental) frequency $1 \leq f_1 \leq 10^6$.

input

```
200 400 800
3100
```

output

```
100
```

Output is 100 because, for the array of 100, 200, 400, 800, and 1600 the sum is 3100 thus the first element of an array is 100.

C. Chess

1 second, 256 megabytes

Johnny has a chessboard with several white pieces already placed on it. He wants to put exactly one black piece on the chessboard and place it in the way it attacks all the white pieces at the same time. He selects between queen, rook, bishop, or knight. Help Johnny find the cheapest piece that can do the job (if we assume values of pieces are ranked as follows: knight < bishop < rook < queen), or conclude that the task is impossible. For the reference to chess pieces movement, please visit: https://www.chess.com/learn-how-to-play-chess#chess-pieces-move.

Input

First line contains an integer t, $1 \leq t \leq 1000$. There will be t test cases, divided by an empty line.

Each test case consists of an 8×8 matrix where each cell is "." or "#", denoting an empty cell and a cell occupied by an enemy figure, respectively.

Output

String, lowercase, one of: "knight", "bishop", "rook", "queen", "impossible" (without quote marks)

input

```
3
........
........
........
.#......
.#......
.#......
........
........

........
..#.....
........
........
.....#..
........
........
........

........
........
..###...
.#......
........
.#......
..#.....
........
```

output

```
queen
knight
impossible
```

In the first example, queen can be placed at A3. Rook at B1 is not enough since it will attack only one piece. Rook at B4 is also forbidden since B4 is already occupied by white piece.

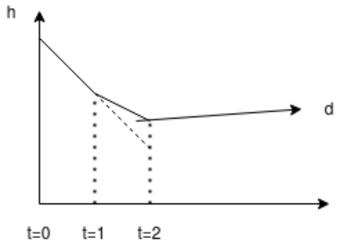
In the second example all the pieces can do the job, but knight (e.g. at D5) is the cheapest one.

The third example is impossible to solve.

D. The End of the World

1 second, 256 megabytes

The Great asteroid is going towards the planet of Earth. It is first spotted in the moment $t = 0$ on the height h. The planet has a defense mechanism that can change the course of the asteroid. Defense system starts to operate at the moment $t = 1$ which is on the height h_2. In every moment t ($t \in N$, step=1) it is possible to apply force to the asteroid and change it's direction angle by a degree d compared to the previous direction angle (see the image). How many times system has to take action in order to save the Earth? Direction of the asteroid has to be parallel or go away from the Earth and it's height must not be below $h = 1$. If the Earth can not be saved, print -1.



Example of the asteroid movement

Input

The first line represents height h of the asteroid when it was first spotted ($1 \leq h \leq 10^6$).

The second line represents height h_2 of the asteroid when defense system starts to operate ($1 \leq h_2 \leq 10^6$).

The third line represents degree d (angle) of direction change ($1 \leq d \leq 90$)

Output

Single integer representing number of times defense system needs to take action in order to save the world. Output is -1 if the world cannot be saved.

input

```
300
299
30
```

output

```
2
```

E. Castle

2 seconds, 256 megabytes

Bob is angrily sitting in a throne at his castle. The source of his bitterness are taxes; Bob is due to pay k euro in taxes and he wants to reduce that amount as much as possible. He has n different tax reliefs at his disposal, each of which consists of two numbers:

1. a_i, the cost of the i-th tax relief, in euro. To use the i-th tax relief, Bob has to buy it first.
2. b_i, the multiplier of the i-th tax relief.

If Bob chooses to use the i-th tax relief, then his tax is influenced as follows:

$$p = k \cdot b_i + a_i$$

Where p is the amount of taxes Bob has to pay *after* he purchases the i-th tax relief, k is the initial amount, a_i is the cost of the i-th tax relief, and b_i is the multiplier which reduces the amount of taxes.

It's also possible to purchase multiple tax relief options, in which case b_i is multiplied. For example, if Bob picks two tax reliefs, i and $i+1$, then he's due to pay:

$$p = k \cdot (b_i \cdot b_{i+1}) + (a_i + a_{i+1})$$

Or, in other words, for each tax relief Bob purchases, his taxes are calculated as follows:

$$p = k \cdot (\ldots \cdot b_{i-1} \cdot b_i \cdot b_{i+1} \cdot \ldots) + (\ldots + a_{i-1} + a_i + a_{i+1} + \ldots)$$

Help Bob find p, the lowest amount of tax he has to pay by choosing the appropriate tax relief options.

The solution is considered correct if the relative or absolute error is lower than 10^{-6}.

Input

The first line contains n, $1 \leq n \leq 256$, the amount of tax relief options Bob can choose from, followed by k, $1 \leq k \leq 10^6$, the amount of taxes Bob has to pay before purchasing tax reliefs.

The next n lines contain two numbers: $1 \leq a_i \leq 256$, the cost of the i-th tax relief, followed by $0 < b_i < 1$, the tax multiplier. The multiplier b_i has at most 6 decimals.

Output

Print p, the lowest amount of tax Bob could pay.

Scoring

(17 points): $1 \leq n \leq 20$.

(83 points): $1 \leq n \leq 256$.

input

```
3 2049
15 0.601
170 0.73
12 0.509
```

output

```
653.807541000000
```

The optimal result for Bob is to buy the first and the third tax relief, in which case, he has to pay:

$$2049 \cdot (0.601 \cdot 0.509) + (15 + 12) = 653.807541$$

If Bob had picked all three tax reliefs, he would have to pay:

$$2049 \cdot (0.601 \cdot 0.73 \cdot 0.509) + (15 + 170 + 12) = 654.569505$$

Which is more costly than if he had picked only the first and third tax relief.

F. Brewer-farmer partnership

3 seconds, 256 megabytes

It is well known that it takes 4 ingredients to make beer: hops, yeast, water and grain. In order to produce good beer, brewers need to get the best ingredients. They already have access to good yeast, grain, and crystal clear water; but the hops in their valley had been ravaged by a terrible disease and could not be grown there anymore. Fortunately, there are a lot of hop farms in a nearby valley, but not all of them are a good fit for every brewer.

In the valley, there are n brewers and n farmers. Brewers keep a tab on each farmers' hop quality and type, so each brewer has a list of all the farmers sorted by preference. Also, farmers prefer brewers who pay for the hops on time, so each farmer has a list of brewers sorted by preference as well.

Brewers and farmers can be considered *happy* if there are no better and available partners they can form a partnership with. In other words, a partnership is *happy* if there are no brewers b_i, b_j, and farmers f_i, f_j such that:

- b_i and f_i are paired;
- b_j and f_j are paired;
- All 4 brewers and farmers (b_i, b_j, f_i, f_j) would be happier if they were arranged into partnerships (b_i, f_j) and (b_j, f_i).

A brewer can form a partnership with only one farmer; the opposite is true as well, meaning that a farmer can only form a partnership with only one brewer.

Combine brewers and farmers so that the brewers get the best possible hops and the farmers are sure that their customer (the brewer) will pay for the hops. In other words, combine brewers and farmers into partnerships so that their cumulative *happiness* is maximized.

Input

The first line contains the number of brewers and farmers n, ($2 \le n \le 1000$). $2 \cdot n$ lines follow.

The first n lines represent the preferences of brewers. Preferences of the i-th brewer are given in i-th line. For each brewer i, there are n numbers given, where each number denotes a farmer, f_j, ($n \le f_j < 2 \cdot n$). The numbers are already sorted and given in descending order by preference, meaning that the first farmer f_j is the most preferred by brewer i.

The next n lines represent the preferences of farmers. Each line represents the preferences of the j-th farmer and contains n numbers, where each number denotes a brewer, b_i, ($0 \le b_i < n$). The numbers are already sorted and given in descending order by preference, meaning that the first brewer b_i is the most preferred by farmer j.

Explanation of example 1:

3 2 $\rightarrow$ Brewer b_0's preference is farmer f_3, while farmer f_2 is less preferred.

2 3 $\rightarrow$ Brewer b_1's preference is farmer f_2, while farmer f_3 is less preferred.

0 1 $\rightarrow$ Farmer f_2's preference is brewer b_0, while brewer b_1 is less preferred.

1 0 $\rightarrow$ Farmer f_3's preference is brewer b_1, while brewer b_0 is less preferred. Result: The optimal result is to make the following pairs:

1. (f_2, b_1)
2. (f_3, b_0)

Output

Print n lines, each containing two numbers (an optimal partnership). The first number in each line should denote the farmer, while the second should denote the brewer.

input
```
2
3 2
2 3
0 1
1 0
```
output
```
2 1
3 0
```

input
```
4
7 4 5 6
4 7 5 6
5 7 6 4
4 5 7 6
3 2 0 1
0 3 1 2
2 0 1 3
0 2 3 1
```
output
```
4 3
5 1
6 2
7 0
```

G. Fibonacci

3 seconds, 256 megabytes

You are given a tree of n nodes, each node labelled with an index k ($1 \le k \le n$). The root node is denoted as $k = 1$.

You're traveling from each node towards the root node. You give a number from the Fibonacci sequence to each node you visit along the way. The starting node gets the first number in the Fibonacci sequence (which is 1), the second node gets the second number of the sequence (which is 1), the third node gets the third number (which is 2) and so on until you reach the root node.

When a node receives a number, it adds it to the number it had previously.

The process is repeated for every node in the tree.

For each node, what's it's the final number? Output it modulo $10^9 + 7$.

Input

The first line contains integer n ($1 \le n \le 3 \cdot 10^5$), the number of nodes.

The next $n - 1$ lines contain two numbers each, u_i and v_i ($1 \le u_i, v_i \le n$), which denote there is an edge between nodes u_i and v_i.

Output

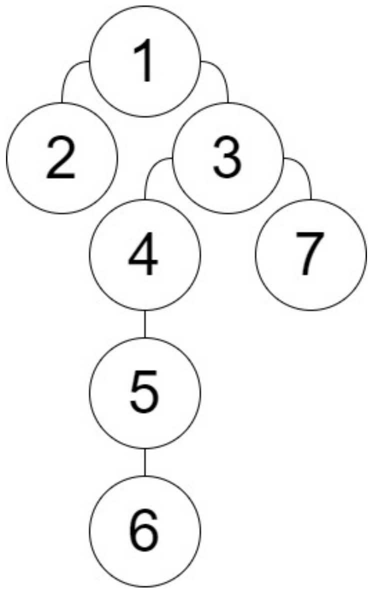
Output n lines. In i-th line output the final number of node with index i.

Scoring

(19 points): $1 \le n \le 100$.

(81 points): $1 \le n \le 2 \cdot 10^5$.

input
```
7
1 2
3 1
4 3
4 5
5 6
7 3
```
output
```
15
1
8
4
2
1
1
```



If we start with node 6, then the travel to node 1 would look like this: 6 -> 5 -> 4 -> 3 -> 1, meaning that node 6 gets fib(1) = 1, node 5 will get fib(2) = 1, node 4 will get fib(3) = 2, node 3 will get fib(4) = 3 and node 1 will get fib(3) = 2.

If we start with node 4, the travel to node 1 looks like this: 4 -> 2 -> 1, so node 4 will get fib(1) = 1, nodel 3 will get fib(2) = 1 and node 1 will get fib(3) = 2. And so on... We will start from each of 7 nodes once.

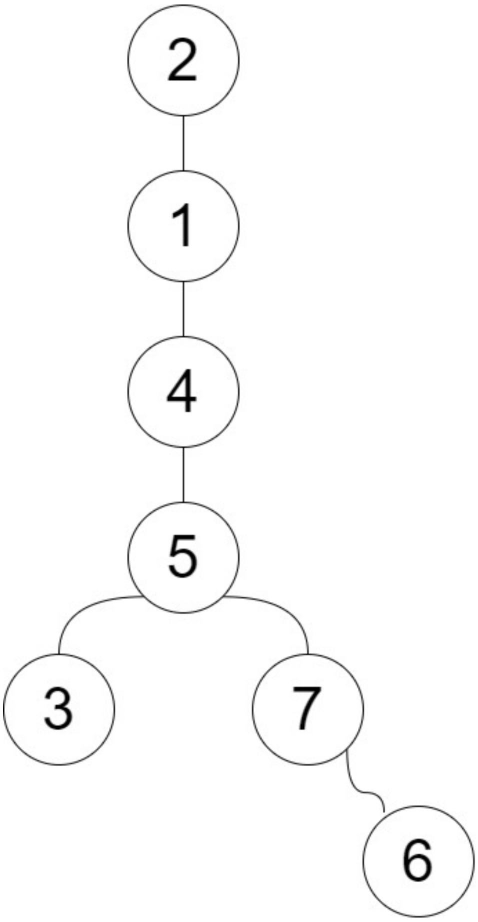
E.g. Node 1 will get: 1 from itself, 1 from node 2, 1 from node 3, 2 from node 4, 2 from node 7, 3 from node 5, and 5 from node 6. It is total 1+1+1+2+2+3+5 = 15. Node 4 will get: 1 from itself, 1 from node 5, and 2 from node 6, which is a total of 4.

The final output is the result for nodes 1,2,3,4,5,6,7 respectively.

H. Coin

2 seconds, 256 megabytes

Alice and Bob are playing on a tree (undirected graph where any two vertices are connected by exactly one path). But it's an unusual game: it involves a coin. At the beginning of the game, the coin is located at the node with index k. In each turn, the player can move the coin to any adjacent node, provided that the coin hasn't visited that node before. The player who cannot move the coin to any adjacent node loses the game. Alice plays first. Assuming both Alice and Bob are playing optimally, determine the winner of the game.



Input

The first line contains integers n ($1 \le n \le 3 \cdot 10^5$) and k ($1 \le k \le n$), the number of nodes and the initial location of the coin, respectively.

The next $n-1$ lines contain the tree's edges. Each line contain numbers a_i and b_i ($1 \le a_i, b_i \le n$), denoting that the i-th edge connects nodes a_i and b_i.

Output

If Alice wins, print `Alice`, otherwise print `Bob`.

input

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

output

```
Bob
```

Coin starts in the node 4.

There are two cases:

1. If Alice moves coin to the node 1, then Bob can move it to node 2, and then Alice can't move it anymore therefore Alice loses in this case.
2. If Alice moves coin to the node 5, then Bob can move it to node 3, and Alice loses. If Bob moved it from 5 to 7, then Bob would lose. However, we assume that both of them play optimally.

In both cases Alice loses, therefore Bob wins.

I. Piles of Twigs

2 seconds, 64 megabytes

Alice and Bob were taking a walk in the forest and, as they were strolling about, they collected twigs of different sizes.

Once they came home, they divided the twigs into three groups: big twigs, medium-sized twigs and small twigs. Now they have three piles of twigs and want to play a game.

In each turn, the player can remove twigs from groups. But, there's a catch, the twig removal is limited to a ruleset which Alice and Bob change each time they play.

First, Alice and Bob come up with the game's rules and they think of n different moves they can make. For each of the n moves, they define k_i, which indicates exactly how many twigs *have* to be removed from the i-th pile in order for the turn to be valid. The player who cannot make a valid move - loses the game.

Players alternate taking turns. Alice plays first, then Bob, then Alice, *et cetera*.

After coming up with the rules, Alice and Bob play q games, each game with different pile sizes.

Both Alice and Bob always play optimally. If Alice plays first, print the winner of the game for each of the q queries.

Input

The first line contains n ($1 \le n \le 10$), the number of rules Alice and Bob come up with.

The following n lines contain three numbers, k_a, k_b and k_c, indicating how many twigs have to be removed from each pile in order for the turn to be valid. ($0 \le k_a \le 30, 0 \le k_b \le 30, 0 \le k_c \le 30$). It is guaranteed that each of the n rules is a unique combination of three numbers.

The following line contains q ($1 \le q \le 2 \cdot 10^5$), the number of games Alice and Bob will play.

The following q lines each contain three numbers; the number of twigs in the first pile a, the number of twigs in the second pile b and the number of twigs in the third pile c, $0 \le a, b, c \le 30$.

Output

For each of the q games, print `Alice` if Alice wins, otherwise print `Bob`.

input

```
3
0 1 0
1 0 0
2 0 7
2
1 1 2
2 0 8
```

output

```
Bob
Alice
```

There are three rules in Alice and Bob's ruleset.

Bob and Alice play two games. In the first game, there is 1 big twig, 1 medium twig and 2 small twigs. Alice play first. In her first turn, she could either take one twig from the big-twig pile or one twig from the medium-twig pile. If she takes a twig from the big-twig file, Bob will take a twig from the medium-twig pile and Alice will lose. If she removes a twig from the medium-twig pile, Bob will remove a twig from the big-twig file and Alice will lose again. Either way, Bob always wins.

In the second game, in her first turn, Alice could take two twigs from the big-twig pile and seven twigs from the small-twig pile. The remaining piles will have 0, 0 and 1 twigs, respectively. This means Bob can take no more valid moves and automatically loses the game.

J. Palenta

3 seconds, 256 megabytes

Alice said she could solve all problems on stem games in less than an hour. Bob told her she should eat much more palenta before claiming such things. Time is running out, and Alice needs your help! Except for moral support, Alice needs you to solve the following problem:

There is $n \times n$ board, and in every cell, there is an integer between 1 and n, inclusive. Any two elements which are in the same row or column are different.

You must select n cells, exactly one from each row and exactly one from each column. Also, every non-selected cell should have either a bigger value than both selected cells in its row and column, or a lower value than both of them.

Also, some of the cells are colored white, while some are colored red. The problem author likes red more than white, so he wants you to select as many red cells as possible while satisfying other requirements. What is the maximal number of red selected cells?

Input

First line contains integer n ($1 \le n \le 200$).

Next n lines contain n integers each, denoting values on the board. Board is a valid board according to the rules stated above.

Next n lines contain n values 0 or 1. If cell $a_{i,j} = 1$ then cell $a_{i,j}$ is red. If it's 0 then $a_{i,j}$ is white.

Output

Output one integer, maximal number of red selected cells.

input

```
3
1 2 3
3 1 2
2 3 1
0 0 1
0 0 0
0 1 0
```

output

```
2
```

In the sample, cells in (row 1, column 3) and (row 3, column 2) are red, while other cells are white.

You can select cells $(1, 3), (2, 1), (3, 2)$. That selects two red cells, thus output is 2.

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform