Problem A
Balatro
Time limit: 12 seconds

You're given some cards arranged in a row. Each card has a value, and is one of two types: add, or multiply.

Compute the score for a row of cards as follows: Initially, the score is zero. Then, process the cards from left to right.

If it is an add card, increase the score by the card's value. If it is a multiply card, multiply the score by the card's value.

The final score is what you get after processing all the cards.

For all possible subsequence sizes, you'd like to know: what is the maximum possible score you can attain for a subsequence of that size (or fewer, if that size isn't possible due to the limit on the number of multiply cards), maintaining their original order? Note, you can't rearrange the cards after choosing them. Also, a subsequence doesn't have to be contiguous.

Input

The first line of input contains two integers n ($2 \leq n \leq 2 \cdot 10^5$) and k ($0 \leq k \leq n$), where n is the number of cards, and k is the maximum number of multiply cards you can use.

Each of the next n lines contains a letter s (s is either '`a`' or '`m`', always lower case) and an integer v ($2 \leq v \leq 1{,}000$), where s indicates whether the card is an add card ('`a`') or a multiply card ('`m`'), and v is the value of the card.

It is guaranteed that the product of all multiply cards is at most 10^9.

Output

Output n lines. On each line, output the maximum score you can achieve for a subsequence of length at most 1, 2, 3, and so on, up to n.

Sample Input 1	Sample Output 1
4 3 a 3 m 2 a 8 m 3	8 24 33 42

Sample Input 2	Sample Output 2
6 1 a 2 m 5 a 2 a 2 m 3 a 3	3 10 13 18 21 21

Problem B
Bikes and Barricades
Time limit: 1 second

Scott wants to ride his bike along a straight road. But the road has some barricades! Scott will ride his bike up to the first barricade and stop.

Model Scott's straight road as the positive Y axis, with Scott starting at the origin. The barricades are line segments, specified by their endpoints. Determine how far Scott can ride, or if his path is completely unobstructed.

Input

The first line of input contains a single integer n ($1 \leq n \leq 1,000$), which is the number of barricades.

Each of the next n lines contains four integers $x1$, $y1$, $x2$ and $y2$ ($-100 \leq x1, y1, x2, y2 \leq 100$, $x1 \neq 0$, $x2 \neq 0$), representing a barricade that runs from $(x1, y1)$ to $(x2, y2)$. It is guaranteed that no barricade will run through the origin.

Output

Output a single real number, which is how far Scott can ride before he hits the closest barricade, or -1.0 if no barricades get in Scott's way. This output will be considered correct if it is within an absolute or relative error of 10^{-2}.

Sample Input 1	Sample Output 1
2 -10 7 5 19 -1 -1 8 21	1.4444444444444446

Sample Input 2	Sample Output 2
2 4 -6 -12 -1 3 5 8 8	-1.0

This page is intentionally left blank.

Problem C
Call for Problems
Time limit: 1 second

The Call for Problems for the ICPC North America Qualifier (NAQ) has finished, and a number of problems were proposed. The judges voted on the difficulty of each problem. The NAQ does not want to be considered an odd contest, so therefore they refuse to use any problem which has an odd number (not divisible by two) for a difficulty rating.

Given the difficulty ratings of the candidate problems, how many were excluded by this rule?

Input

The first line of input contains a single integer n ($1 \leq n \leq 50$), which is the number of candidate problems.

Each of the next n lines contains a single integer d ($0 \leq d \leq 100$), which are the difficulty ratings of the n problems.

Output

Output a single integer, which is the number of candidate problems excluded by the rule.

Sample Input 1	Sample Output 1
2 2 3	1

This page is intentionally left blank.

Problem D
Colorful Trees
Time limit: 1 second

Given a tree with colored vertices, for each edge, how many pairs of vertices with the same color have that edge on the path between them? Note that since it's a tree, each pair of nodes has exactly one path between them.

Input

The first line of input contains a single integer n ($2 \leq n \leq 10^5$), which is the number of nodes in the tree. The nodes are numbered from 1 to n.

Each of the next n lines contains a single integer c ($1 \leq c \leq n$). These are the colors of the nodes, in order.

Each of the next $n-1$ lines contains two integers a and b ($1 \leq a < b \leq n$), denoting an undirected edge from node a to node b.

Output

Output $n-1$ lines. On each line, output a single integer, which is the number of pairs of vertices with the same color that have that edge on the path between them. Output these answers for the edges in the order that they appear in the input.

Sample Input 1	Sample Output 1
6 3 1 2 1 2 2 2 6 4 5 1 4 3 4 1 2	2 2 3 2 3

Sample Input 2	Sample Output 2
4 2 2 2 2 3 4 2 4 1 2	3 4 3

Problem E
Dishonest Lottery
Time limit: 1 second

You suspect your local lottery of cheating! Some numbers are coming up too often!

Each week, the lottery randomly selects five numbers in the range from one to fifty. So, each number should appear about 10% of the time if the numbers are truly chosen randomly. If a number appears far more than that, it's suspicious!

Determine if a set of lottery drawings is suspicious by listing all numbers that appear too often. To allow for random error, you'll need to flag any number that appears more than 20% of the time.

Input

The first line of input contains a single integer n ($1 \leq n \leq 1{,}000$). There will be $10 \cdot n$ lottery results for you to analyze.

Each of the next $10 \cdot n$ lines contains 5 integers x ($1 \leq x \leq 50$). Each line represents a drawing. All values on a line are unique.

Output

On a single line, output all numbers that appear strictly more than $2 \cdot n$ times in the list. If there is more than one, output them space-separated, in sorted order from smallest to largest. If there aren't any, output -1.

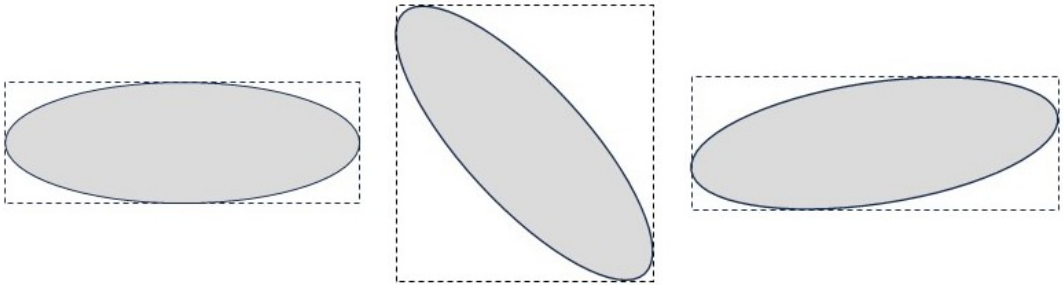
Sample Input 1	Sample Output 1
1 32 30 16 45 27 34 45 35 31 42 1 12 26 50 13 34 50 36 21 39 47 7 41 18 45 28 48 2 8 4 16 40 17 2 19 50 4 30 15 6 31 13 33 46 18 49 23 24 17 48	45 50

This page is intentionally left blank.

Problem F
Ellipse Eclipse
Time limit: 1 second

An *Ellipse* is a 2D geometric figure. It consists of the set of all points such that the sum of the distance from each point to two specific points is constant. The two specific points are called the *Foci* of the ellipse. The *Major Axis* of an ellipse is the diameter of the ellipse that runs through both foci. It is the longest diameter of the ellipse.

Given an ellipse's two foci and the length of its major axis, determine the coordinates of the tightest axis-aligned bounding box that can block out that ellipse.



Input

The single line of input contains five integers x_1, y_1, x_2, y_2 ($-100 \leq x_1, y_1, x_2, y_2 \leq 100$) and a ($\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \leq a \leq 1{,}000$), where the two foci of the ellipse are at (x_1, y_1) and (x_2, y_2) and the length of the major axis is a (Note that's the major axis, not the semi-major axis or the major radius).

Output

Output four space-separated real numbers on a single line. The numbers, in order, are x_{low}, y_{low}, x_{high}, y_{high}, where $(x_{\text{low}}, y_{\text{low}})$ is the lower left corner of the tightest bounding box of the ellipse, and $(x_{\text{high}}, y_{\text{high}})$ is the upper right corner of the tightest bounding box of the ellipse. Each output will be considered correct if it is within an absolute or relative error of 10^{-4}.

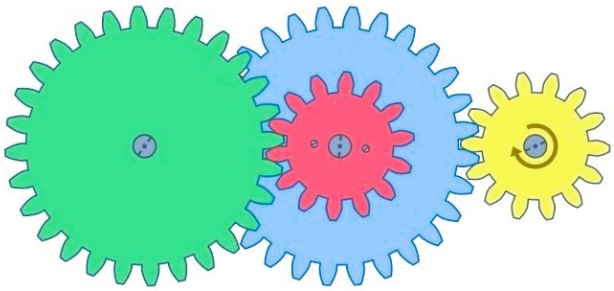
Sample Input 1	Sample Output 1
-5 0 5 0 16	-8.000000 -6.244998 8.000000 6.244998

Sample Input 2	Sample Output 2
51 23 19 67 70	7.778685 13.871235 62.221315 76.128765

This page is intentionally left blank.

Problem G
Gears and Axles
Time limit: 1 second

You have an assortment of circular gears with varying numbers and sizes of teeth. You also have a motor that spins at one revolution per second, as well as an unlimited number of (identical, arbitrarily long) axles. The motor and all gears fit the axles, and everything attached to a particular axle rotates at the same angular speed. Two gears with the same size teeth can be enmeshed with each other. Gears with different size teeth cannot be enmeshed with each other (though they can be placed on the same axle).



You can arrange the gears and axles in any order. What is the maximum rate at which the last gear/axle in sequence spins that you can achieve? Because this may be large, output the *natural log* of the value.

Input

The first line of input contains a single integer n ($0 \leq n \leq 10^5$) denoting the number of gears.

Each of the next n lines contains two integers s ($1 \leq s \leq 10^5$) and c ($3 \leq c \leq 10^5$), one for each gear in your collection, where s is the size of the teeth of the gear and c is the count of the number of teeth.

Output

Output a single line with a single number equal to the *natural log* of the maximum angular speed you can achieve with your motor and axles and gears in your collection. This output will be considered correct if it is within an absolute or relative error of 10^{-6} .

Sample Input 1

```
6
19 364
21 1023
19 66
19 242
21 807
19 675
```

Sample Output 1

```
2.9704451880078357
```

Sample Input 2

```
4
33 10
33 27
44 10
44 27
```

Sample Output 2

```
1.9865035460205664
```

Problem H
Genetic Reconstruction
Time limit: 4 seconds

You're in charge of studying a new species, and you'd like to make sure the work that you've collected is correct.

There are several creatures. For each one, you know the eye color they have. This is denoted by one of the first 20 English lowercase letters (from '**a**' to '**t**').

You think that there is a gene that controls eye color. Your hypothesis is that each creature has two *Alleles*. An allele is represented by a lowercase English letter. The resulting eye color of the creature is the allele of the two which comes first alphabetically.

In addition, you've identified two parents of each creature. Some parents' information may be missing; either both parents' information will be available or both will be missing. A creature inherits one allele from each parent; any of the four combinations is possible. For example, one parent with alleles '**ak**' (thus eye color '**a**'), and another with alleles '**em**' (eye color '**e**'), might have a child with alleles '**ae**' (eye color '**a**'), '**am**' (eye color '**a**'), '**ek**' (eye color '**e**') or '**km**' (eye color '**k**').

Given the number of creatures, the parent information, and the eye color of each creature, determine if this information is consistent with your hypothesis.

Input

The first line of input contains a single integer n ($1 \leq n \leq 20$), which is the number of creatures in your study. The creatures are numbered from 1 to n.

Each of the next n lines contains two integers p_1, p_2 (($1 \leq p_1, p_2 \leq n, p_1 \neq p_2$) or $p_1 = p_2 = 0$) and a character c ($c \in \{$'**a**', ... ,'**t**'$\}$), where p_1 and p_2 represent the parents of the given creature (or are both 0 if the parents are unknown), and c is the given creature's eye color. Note that either both parents will be known or both will be unknown, and that both parents' numbers will be less than the number of the given creature; no creature can be its own ancestor or descendant.

Output

If the information is consistent, output the alleles for each creature one per line, with no spaces; otherwise output -1. If there are multiple possible solutions, output only the one that comes first alphabetically (e.g., the alphabetically first allele pair for first creature, then second, and so on).

Sample Input 1

```
3
0 0 a
0 0 b
1 2 c
```

Sample Output 1

```
ac
bc
cc
```

Sample Input 2

```
3
0 0 c
0 0 c
2 1 a
```

Sample Output 2

```
-1
```

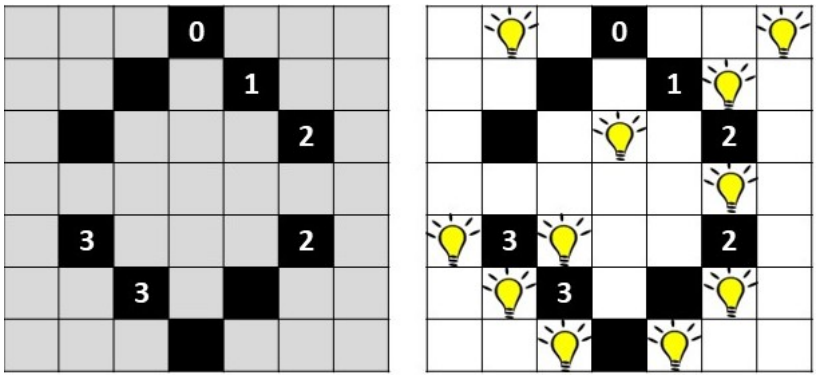
Problem I
Light Up
Time limit: 1 second

Light Up is a pencil puzzle. Your job will not be to play Light Up, but simply to judge whether a player's solution is correct.

The game is played on a square grid. Some of the cells are blocked, and some of the blocked cells have a number from 0 to 4. The player must place light bulbs in open cells. Each light bulb can light all of the open cells above, below, left, and right (but not diagonally) until the light reaches the edge of the grid or a blocked cell. The player must place light bulbs so that:

- Every open cell is lit.
- No two light bulbs can shine on each other.
- Any blocked cell with a number in it must have exactly that number of light bulbs immediately adjacent above, below, left, or right. Diagonals do not count.

The following is an example grid with its solution:



Given a grid with light bulbs placed, determine whether it is, in fact, a solution. Note that if a grid has no open cells, and does not violate any other constraints, it is trivially solved.

Input

The first line of input contains a single integer n ($1 \leq n \leq 30$), which is the number of rows and columns in the grid.

Each of the next n lines contains exactly n characters from the set {'`.`','`X`','`?`','`0`','`1`','`2`','`3`','`4`'}. This is the grid, with '`.`' representing an open cell, '`X`' representing a blocked cell, '`?`' representing a light bulb, and the numbers '`0`','`1`','`2`','`3`','`4`' representing a blocked cell with a constraint on the number of adjacent light bulbs.

Output

Output a single integer, which is 1 if the input is a valid solution, and 0 otherwise.

Sample Input 1

```
7
.?.0..?
..X.1?.
.X.?.2.
.....?.
?3?..2.
.?3.X?.
..?X?..
```

Sample Output 1

```
1
```

Sample Input 2

```
7
.?.0..?
..X.1?.
.X...2.
.....?.
?3?..2.
.?3.X?.
..?X?..
```

Sample Output 2

```
0
```

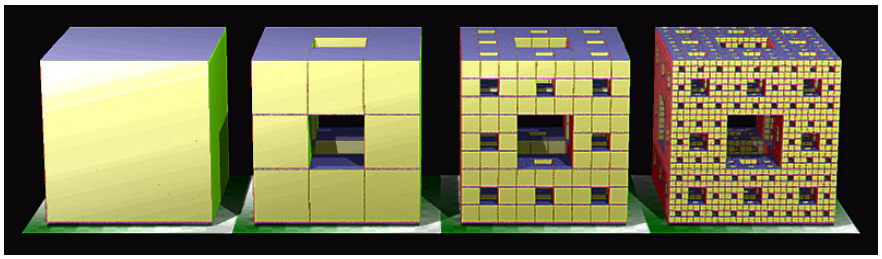
Problem J
Menger Sponge
Time limit: 2 seconds

The Menger sponge is a simple 3D fractal. Its level-L approximation can be constructed with the following algorithm:

> Start with a single solid $1 \times 1 \times 1$ cube with opposite corners at $(0, 0, 0)$ and $(1, 1, 1)$.
> For each iteration $i = 1, \ldots, L$:
>
> > For each cube:
> >
> > > Cut the cube into a regular $3 \times 3 \times 3$ grid of 27 subcubes.
> > > Delete the seven subcubes that don't touch an edge of the parent cube (see illustration).

The points in the level-L Menger sponge are those that remain after running the above algorithm. Points exactly on the boundary of cubes that remain in the sponge **are** part of the sponge.

The picture below shows the result for $L = 0$ through $L = 3$:



Given a level L and a point in space given by three rational coordinates, determine if the point is in the level-L Menger sponge.

Input

The single line of input contains seven integers L, x_{num}, x_{denom}, y_{num}, y_{denom}, z_{num}, z_{denom}:

$$0 \leq L \leq 10^5$$

$$0 < x_{\text{num}} < x_{\text{denom}} \leq 10^6$$

$$0 < y_{\text{num}} < y_{\text{denom}} \leq 10^6$$

$$0 < z_{\text{num}} < z_{\text{denom}} \leq 10^6$$

where L is the level of the Menger Sponge and the point in question is $\left(\frac{x_{\text{num}}}{x_{\text{denom}}}, \frac{y_{\text{num}}}{y_{\text{denom}}}, \frac{z_{\text{num}}}{z_{\text{denom}}}\right)$.

Output

Output a single integer, which is 1 if the point is in the level-L Menger Sponge, or 0 if not.

Sample Input 1	Sample Output 1
1000 1 3 1 3 1 3	1

Sample Input 2	Sample Output 2
2 49 81 5 6 20 81	1

Sample Input 3	Sample Output 3
3 49 81 5 6 20 81	0

Problem K
Rhythm Flow
Time limit: 1 second

You are designing a scoring algorithm for the new hit rhythm game *Rhythm Flow* where players must press a button in time to the music. During a round of Rhythm Flow, there are points in time when a visual indicator flashes on the screen. Players are *expected* to press the button at those times (and only at those times). However, since human reaction time is not instantaneous, the game gives the player some leeway and accepts a button press a few milliseconds earlier or later than expected. More accurate presses are worth more points.

The following table lists how many points a player gets depending on how far away the *actual* button press is from the *expected* button press, in milliseconds:

Time Difference (ms)	Points
$[0, 15]$	7
$(15, 23]$	6
$(23, 43]$	4
$(43, 102]$	2

Wildly inaccurate presses with a difference of 103 milliseconds or more score no points.

During gameplay, the player presses the button some number of times. To score the game, match each actual button press with at most one expected button press, with the following restriction: if one actual button press happens before another actual button press and both button presses are matched with expected button presses, then the expected button press for the first must be strictly before the expected button press for the second.

Because you are generous, you decide to compute the matching that maximizes the number of points the player earns. Compute the final score of the round of Rhythm Flow under this matching.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 2{,}000$), where n is the number of expected button presses and m is the number of actual button presses.

Each of the next n lines contains a single integer e ($1 \leq e \leq 10^9$). These are the times of the expected button presses in milliseconds from the start of the round. It is guaranteed that these values are unique and in sorted order.

Each of the next m lines contains a single integer a ($1 \leq a \leq 10^9$). These are the times of the actual button presses in milliseconds from the start of the round. It is guaranteed that these values are unique and in sorted order.

Output

Output a single integer: the total number of points the player earns given a maximally-generous scoring engine.

Sample Input 1

```
3 4
100
200
300
99
201
240
323
```

Sample Output 1

```
20
```

Sample Input 2

```
4 3
1000
2000
2500
3000
1103
2598
4000
```

Sample Output 2

```
2
```

Sample Input 3

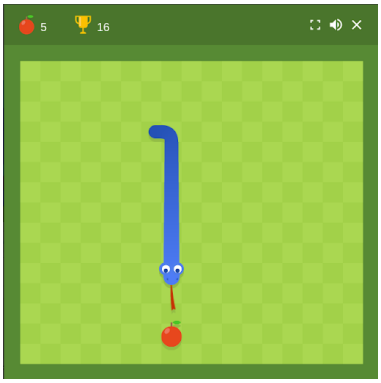
```
2 2
100
144
56
100
```

Sample Output 3

```
7
```

Problem L
Snake
Time limit: 2 seconds

Snake is a video game classic, preserved at the Museum of Modern Art (MoMA) and listed as one of the "Top 100 Video Games" of all time. The goal of the game is to move a snake's head to an apple. Once the snake reaches the apple, it eats it and grows in length. A new apple is placed, which the now grown snake must then eat.



Google's version of Snake

The game is played on a grid, and every segment of the snake's body occupies one cell. The snake's head can turn in three directions, but it cannot go backwards. The body follows the head. The head may not collide with the body or exit the grid. Since the entire snake moves at the same time, the head is allowed to enter the cell that the tail is vacating.

Playing the game requires quickness and foresight. It's all too easy to take turns that put the snake head in a position where it's doomed to hit the wall or its body before reaching the apple, especially as the snake grows longer.

You're being asked to write a program that can determine whether the snake's head can reach the apple from a given position, or not and the snake is doomed to die.

Input

The first line of output contains two integers r and c ($1 \le r, c \le 10, r \cdot c \ge 2$), where the grid has r rows and c columns.

Each of the next r lines contains a string of length exactly c characters from the set

{'.','0', ... ,'9','a', ... ,'f','A'}

where '.' represents an open cell in the grid, the hexadecimal digits '0', ... ,'9' and 'a', ... ,'f' represent the snake, and 'A' represents the apple. The snake may be anywhere from one to sixteen characters long, with '0' as its head, followed by the other hexadecimal digits in strict order ('1' follows '0', '2' follows '1', etc., with no skipping digits.). It is guaranteed that there is at most one of each digit, each digit (except '0') is adjacent to the immediately previous digit, and that there is exactly one apple in the grid.

Output

Output a single integer, which is 1 if the snake can reach the apple, and 0 if it cannot and is doomed to die.

Sample Input 1

```
5 8
......01
....98.2
...A.7.3
.....654
........
```

Sample Output 1

```
1
```

Sample Input 2

```
5 4
...A
....
6789
5432
..01
```

Sample Output 2

```
0
```

Sample Input 3

```
5 5
....A
.....
678..
54321
....0
```

Sample Output 3

```
1
```

Sample Input 4

```
4 4
567A
4389
12ba
0dc.
```

Sample Output 4

```
1
```